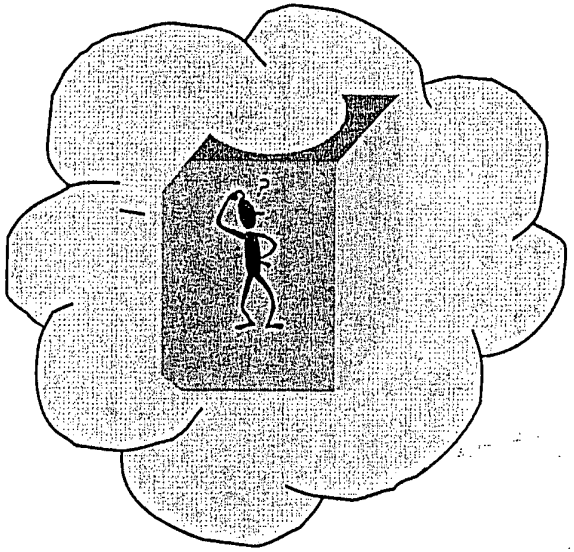
TODAY



- Tools restricted to a specific data source
- Difficulty in analyzing data from various data sources using common tools
- Stove-Pipe systems that are costly to enhance
- Inability to collaborate on multiple data sources at the same time to solve a problem

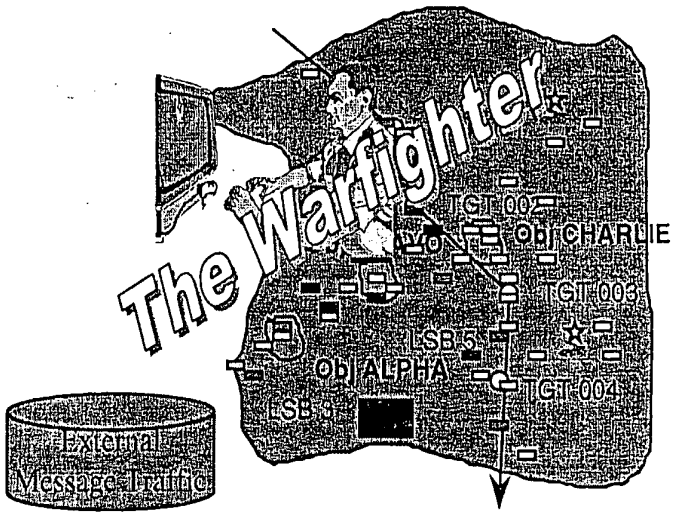


Fig. 1

Collaborative Interoperable Environment

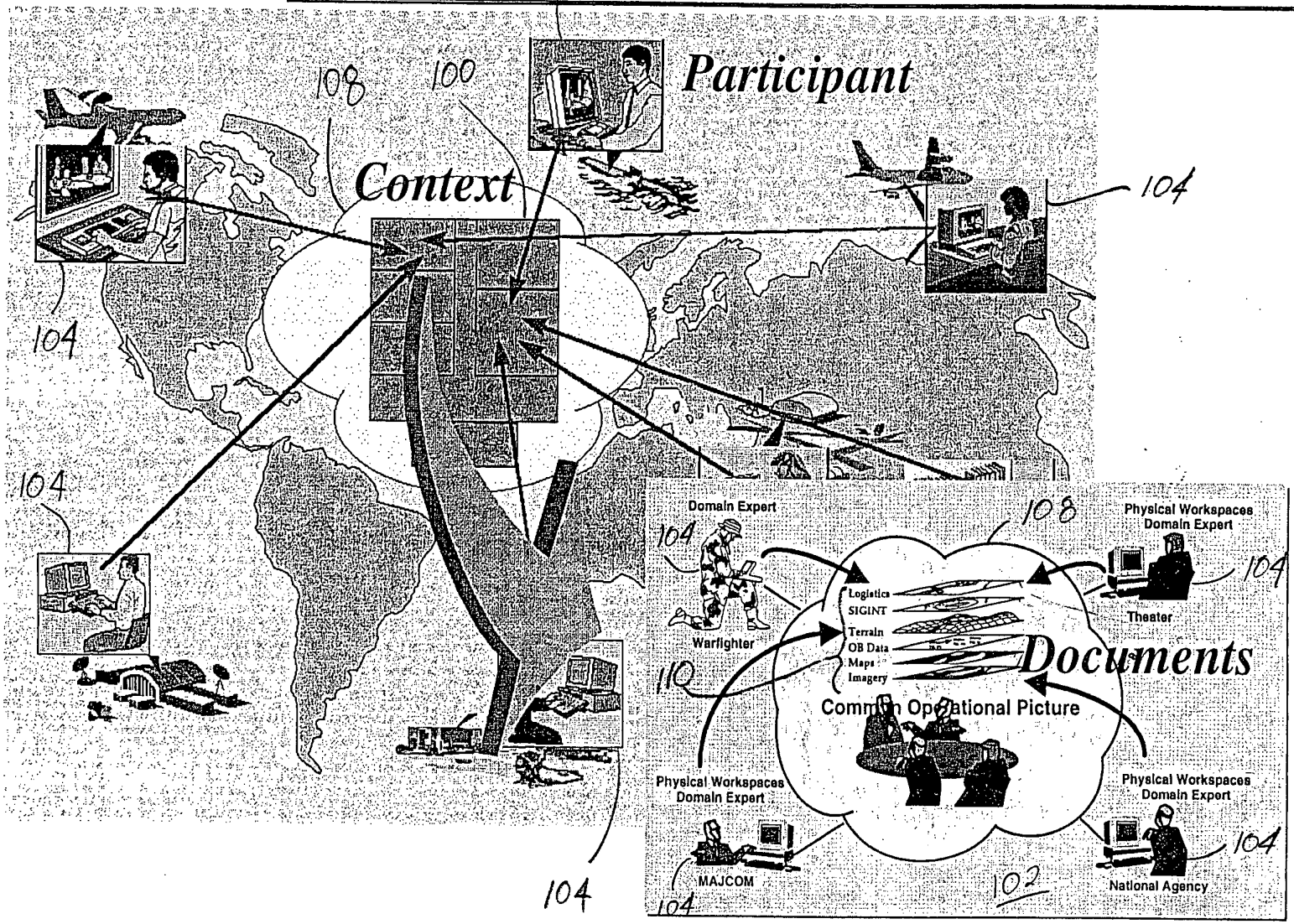
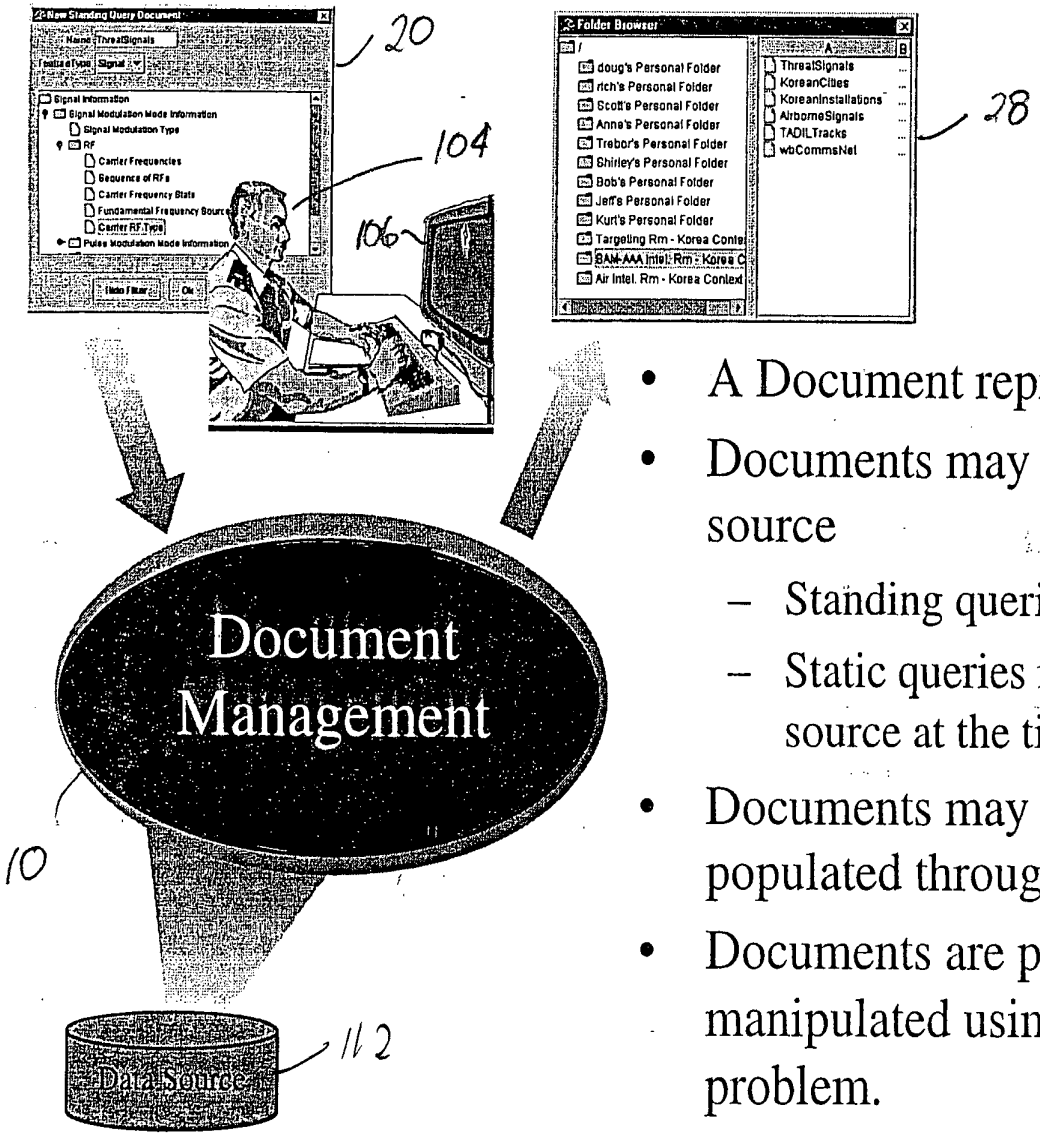


Fig 2

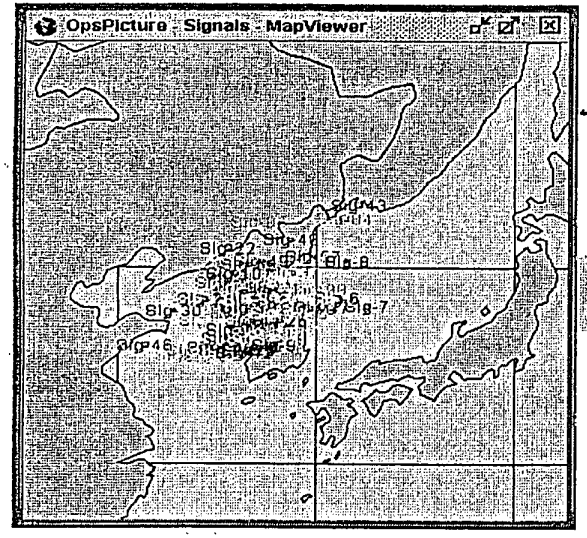
"Document" based data manipulation



- A Document represents a collection of data
- Documents may be created by querying a data source
  - Standing queries are constantly evaluated
  - Static queries represent the state of the data source at the time the query was initiated
- Documents may initially be empty and populated through user or agent actions
- Documents are placed in conferences to be manipulated using the tools that best solve the problem.

Fig 3

Thin Clients interact with data represented by a document



18

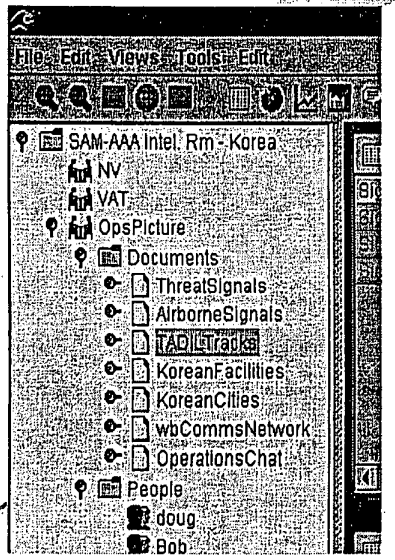
10

Domain Data Is Extended For Visualization and Control

- Visualization and control properties (e.g., color, selection, symbol, etc.) become part of the data
- Client viewers focus on presentation of information
- Clients do not require logic dealing with collaboration
- Clients do not require complex logic to access data

104

108

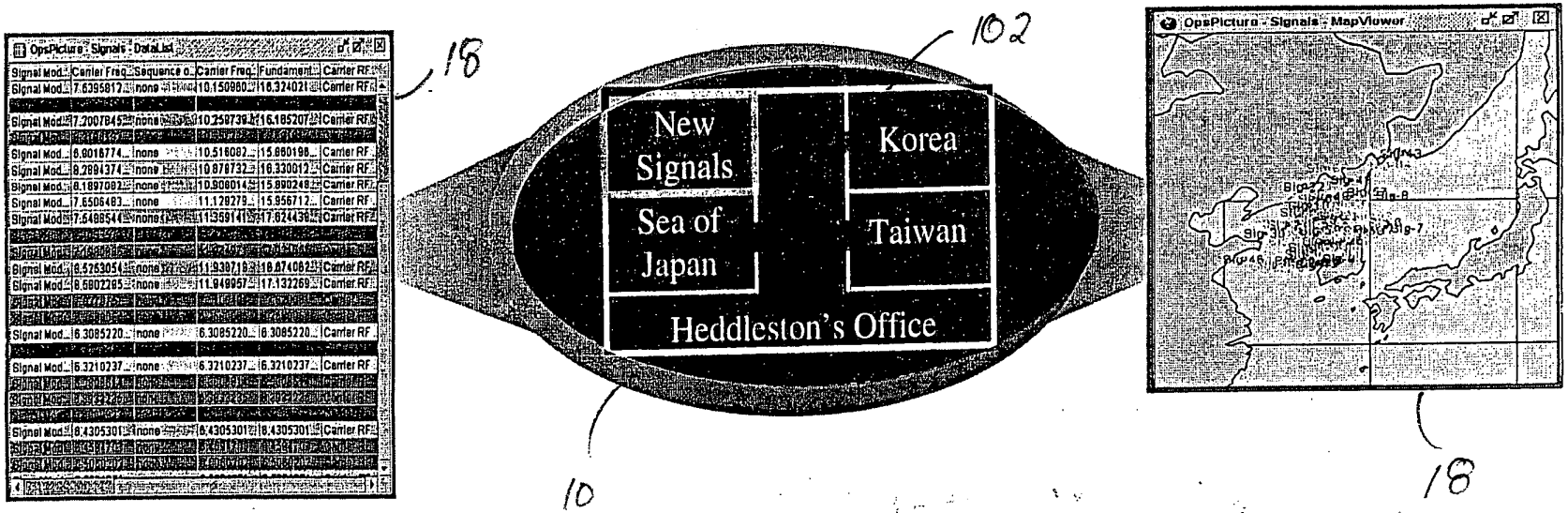


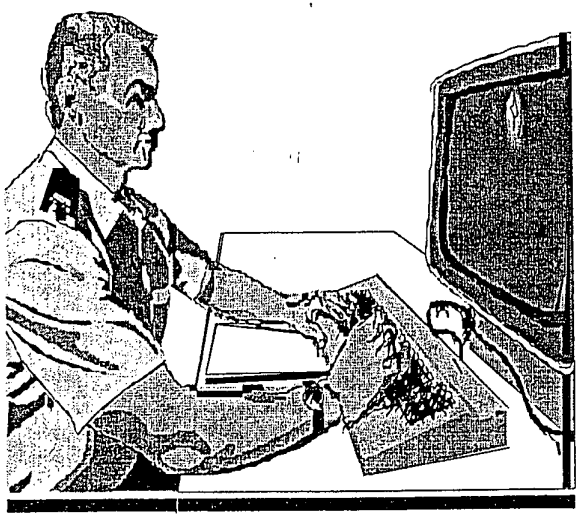
60

Displaying documents using various tools

Fig 4

Collaboration on Multiple Views

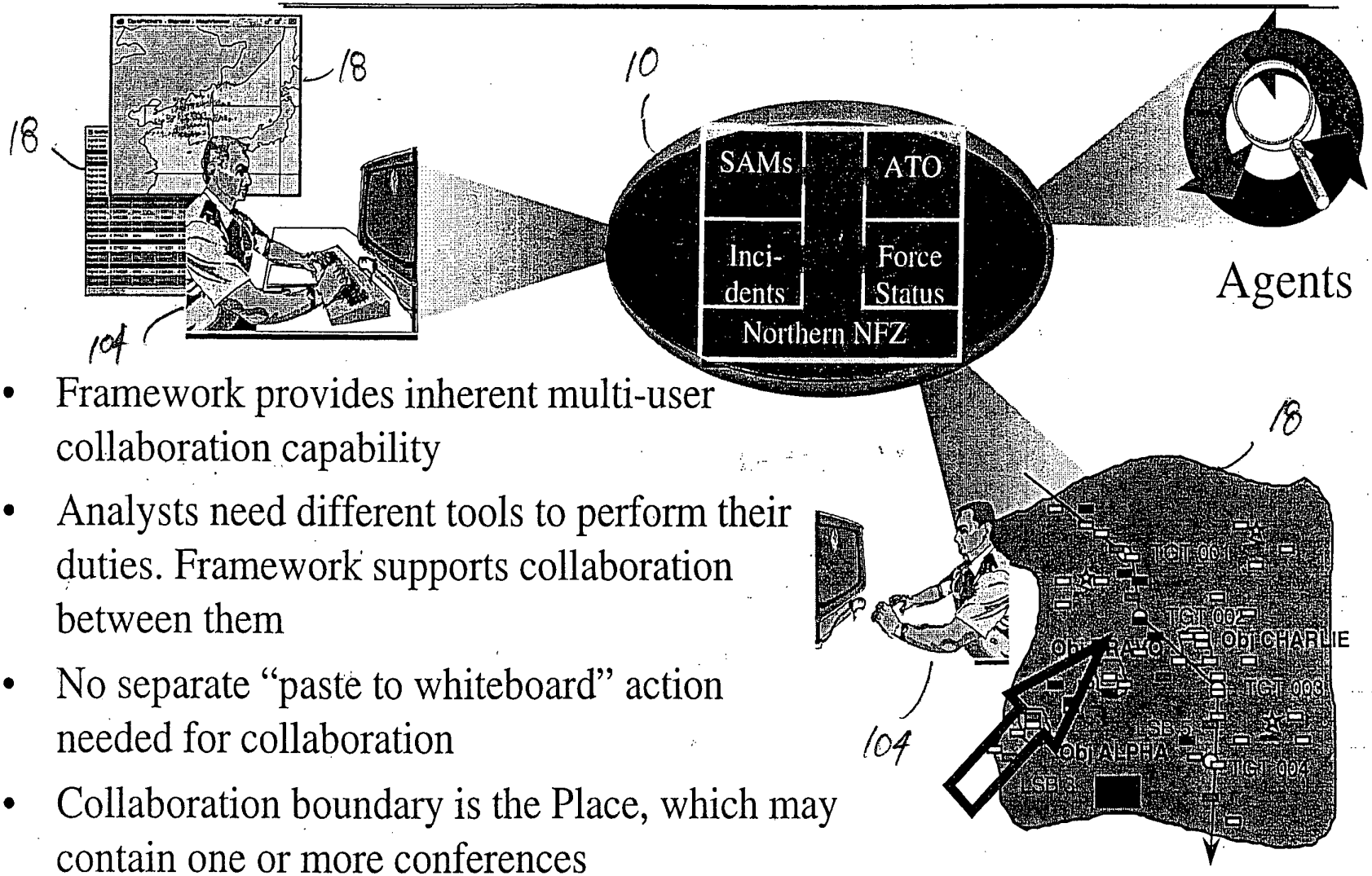




- Single user collaboration
- Multiple tools in the same conference coordinate visualization (e.g. highlight, color)
- All tools in a conference cooperate for problem solving
- No tool-to-tool communication

Fig 5

Flexibility and Collaboration



- Framework provides inherent multi-user collaboration capability
- Analysts need different tools to perform their duties. Framework supports collaboration between them
- No separate "paste to whiteboard" action needed for collaboration
- Collaboration boundary is the Place, which may contain one or more conferences
- Collaborators may be agents as well as humans

Fig 6

Collaboration Summary

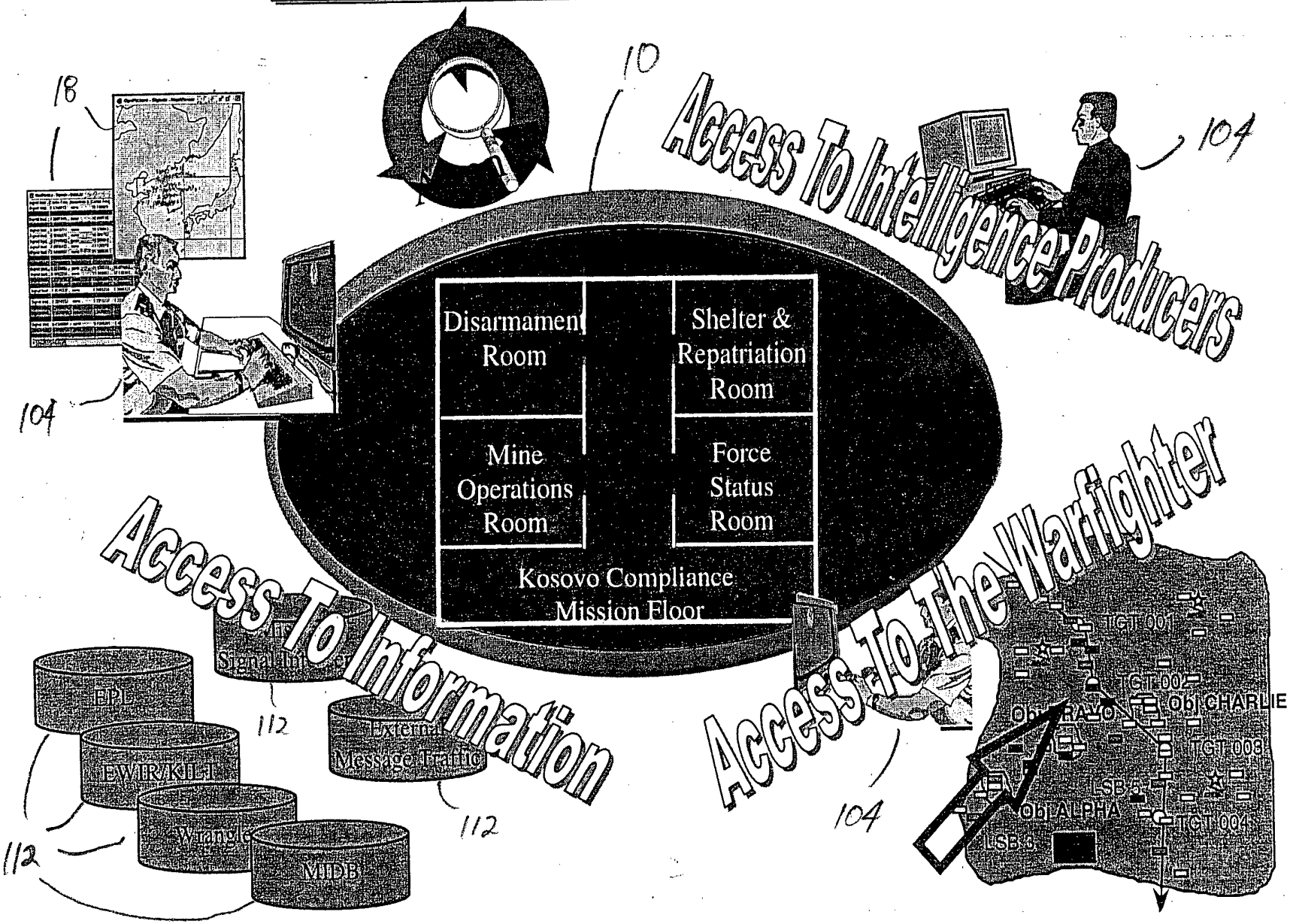


Fig. 7

Architectural Strategy

Key Reference Architectures
- Object Management Architecture (OMA)
  - OpenGIS, CosServices
- COE Layered Architecture
- UCA Cryptologic Framework
- USIGS
  - GIAS

Key Data Models
- SOM, MIDB, JCDB, ASAS, L245, ECDS, TEXTA

Architectural Patterns
- Layered Architecture
- Data Centric Architecture
  - — Information Management Framework
  - — Interactive Analysis Framework
- Mission Management Architecture
  - — Task Management Framework
  - — Resource Management Framework

COTS SW Infrastructure
- JAVA/C++
- CORBA
- Enterprise Java Beans
- RDBMS/ODBMS
- Microsoft Windows
- WEB Server/Browser
- XML / DOM

COTS HW
- UNIX SMP Server
- NT Workstations

Fig 8

Services Based Architecture

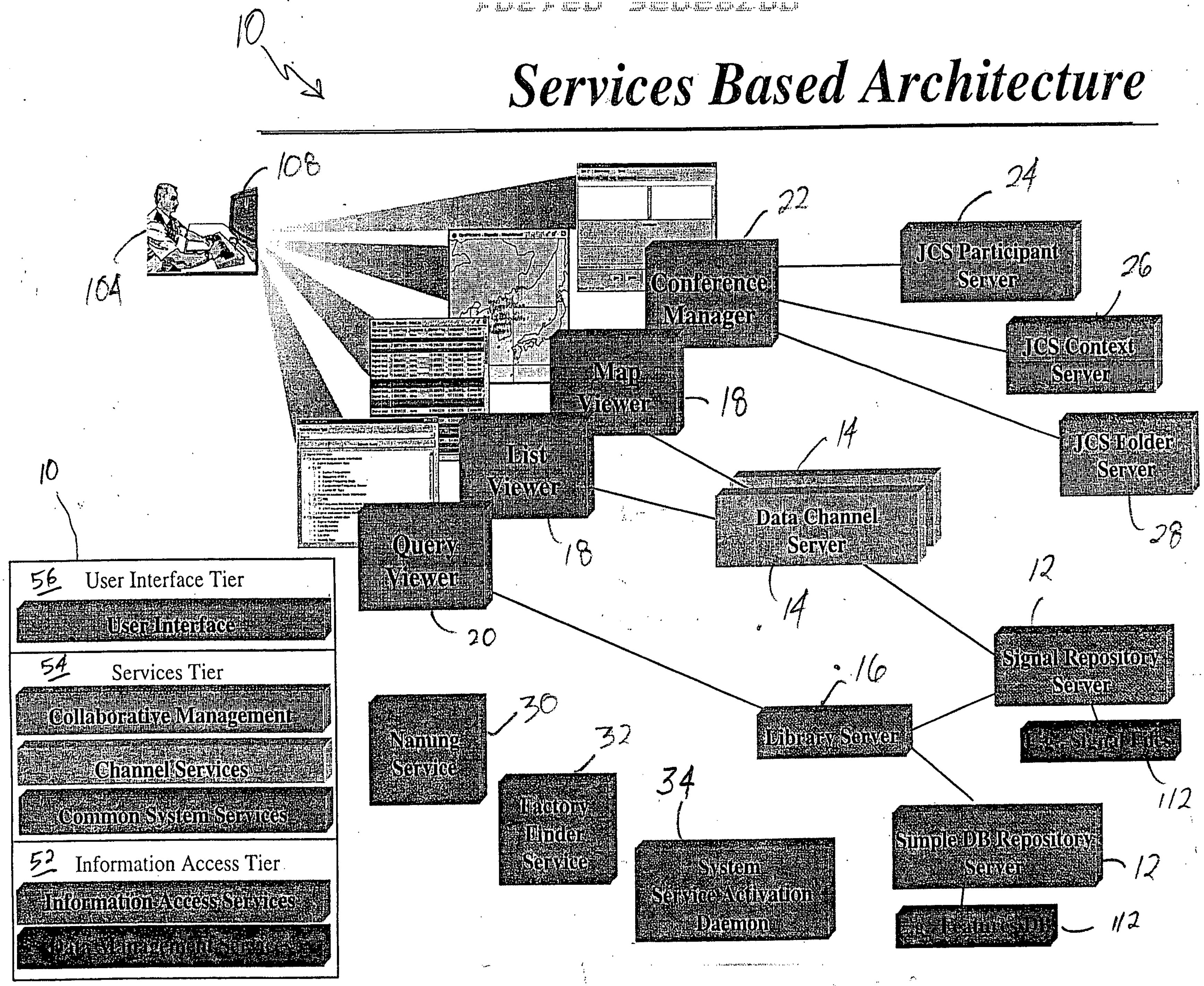

10
104
108
22
Conference Manager
24
JCS Participant Server
26
JCS Context Server
28
JCS Folder Server
18
Map Viewer
18
List Viewer
20
Query Viewer
14
Data Channel Server
14
16
Library Server
12
Signal Repository Server
112
12
Simple DB Repository Server
112
30
Naming Service
32
Factory Finder Service
34
System Service Activation Daemon
10
56 User Interface Tier
User Interface
54 Services Tier
Collaborative Management
Channel Services
Common System Services
52 Information Access Tier
Information Access Services

Fig. 9

Extending The Infrastructure

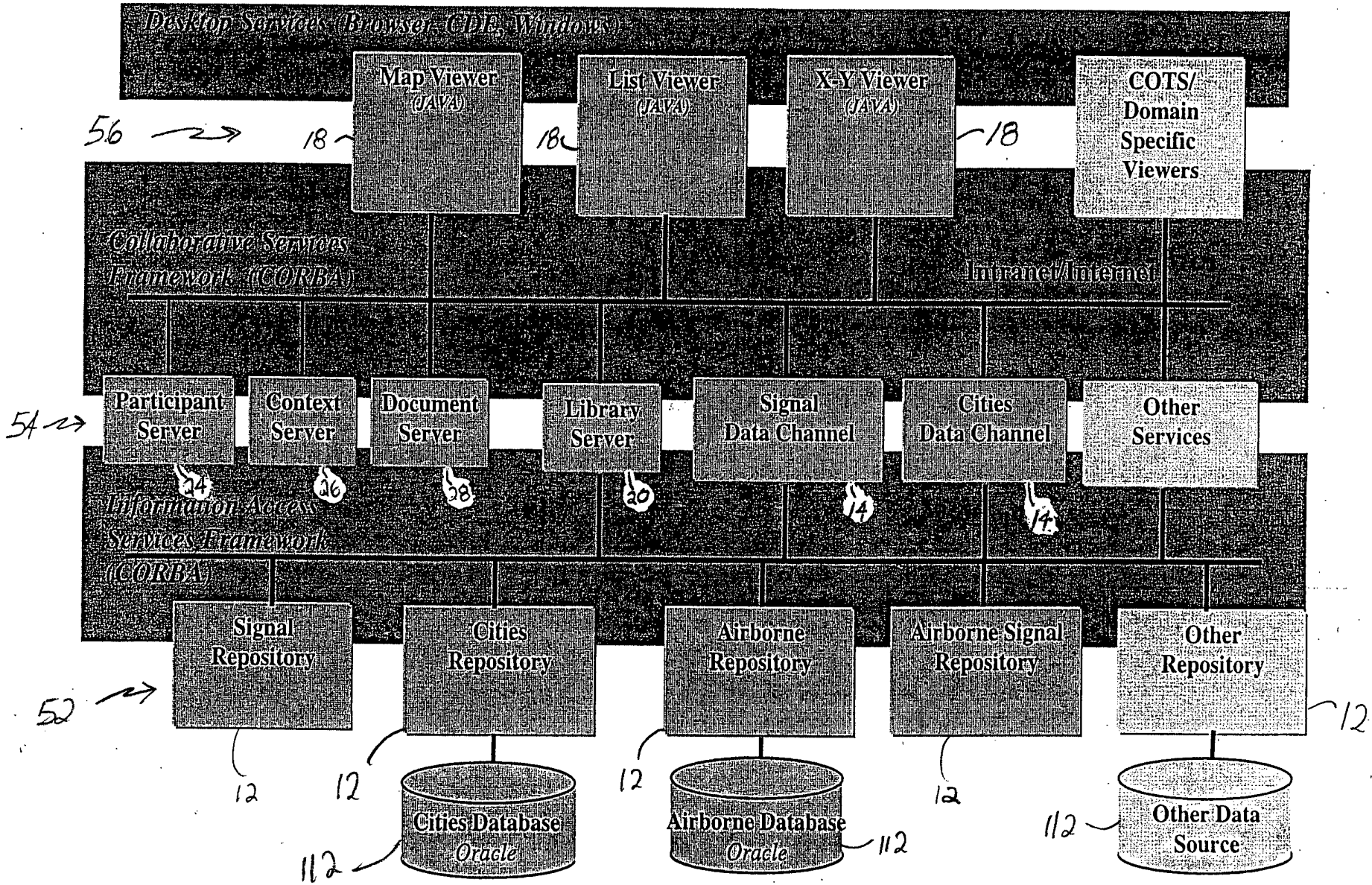
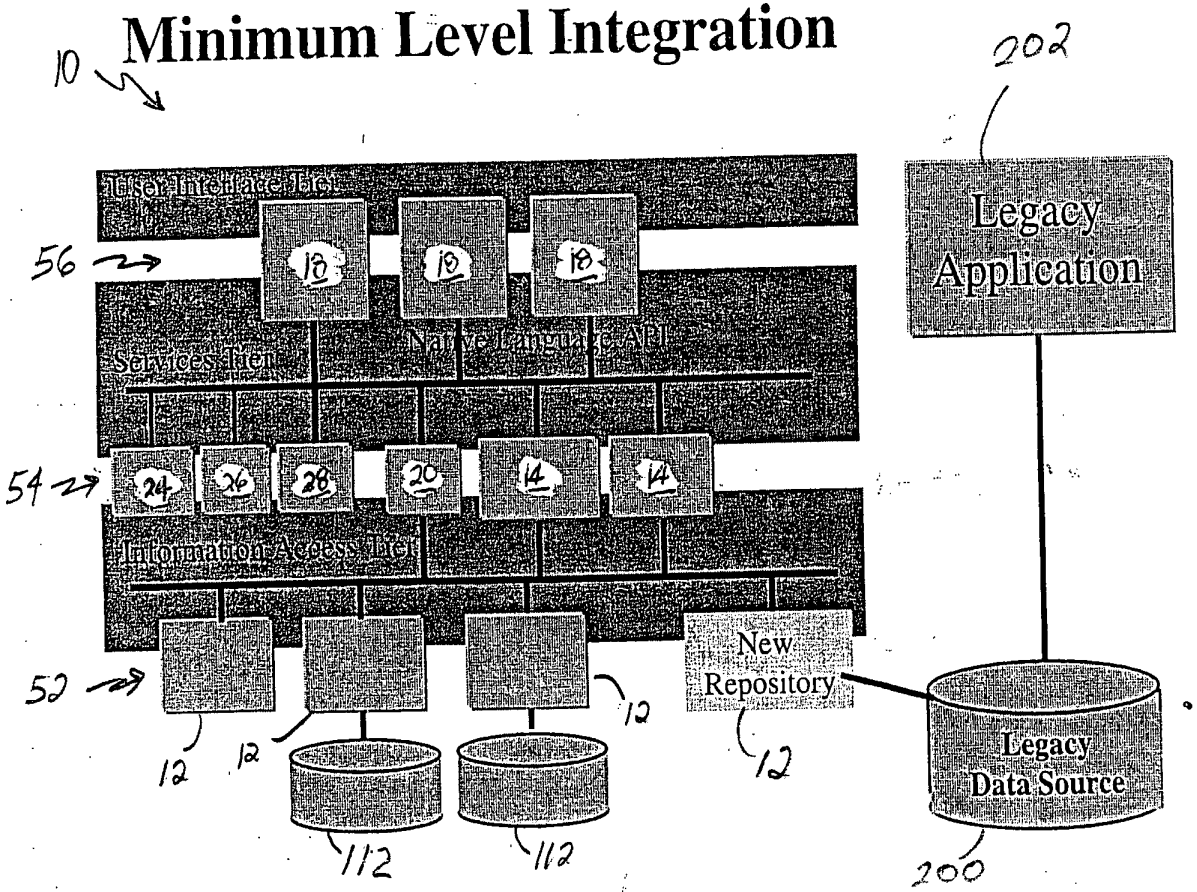


Fig. 10

Integration with legacy systems

Minimum Level Integration

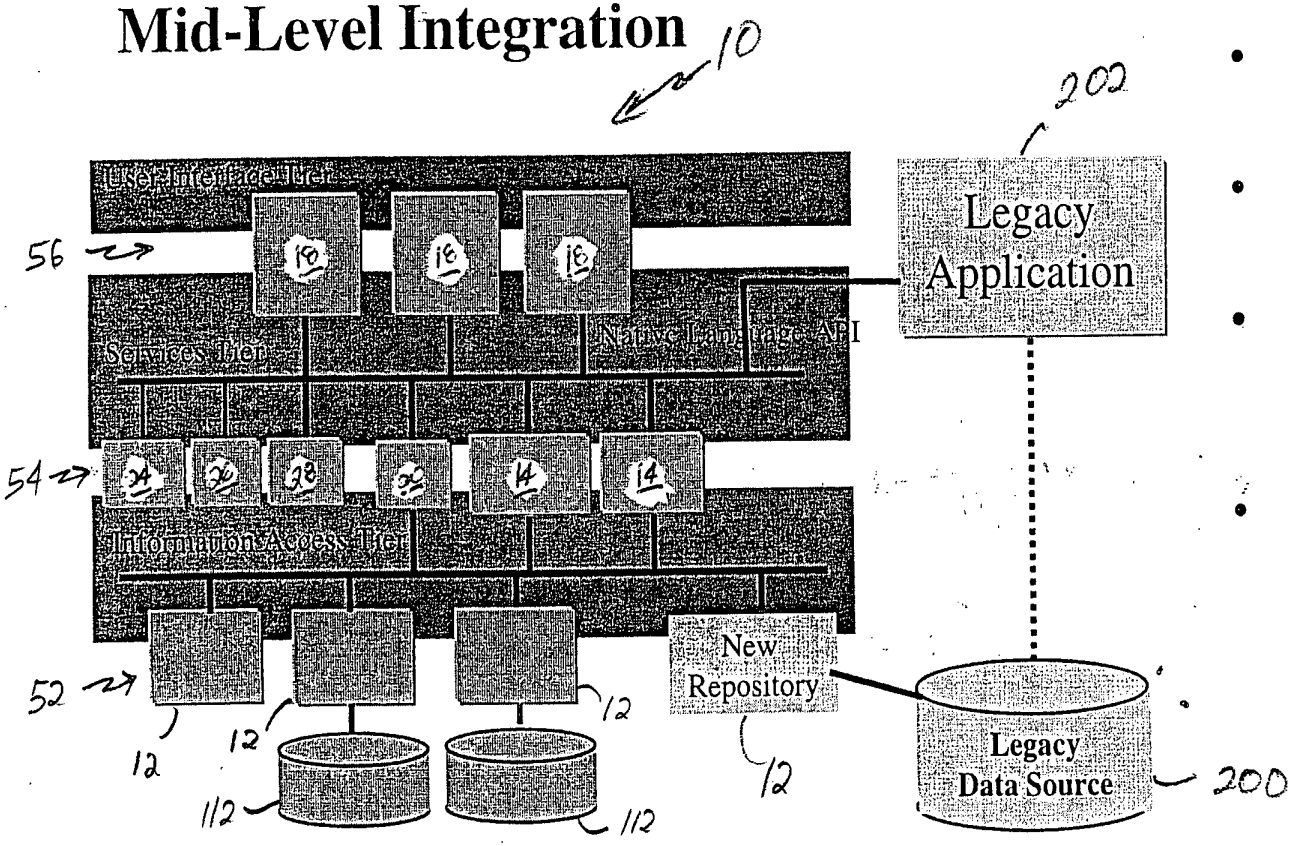


- Provide access to legacy data source through a new repository
- No legacy software changes required
- New data source is available for collaborative processing
- Provides new options for extending system capabilities
- Low/No Risk implementation

Fig 11

Integration with legacy systems

Mid-Level Integration



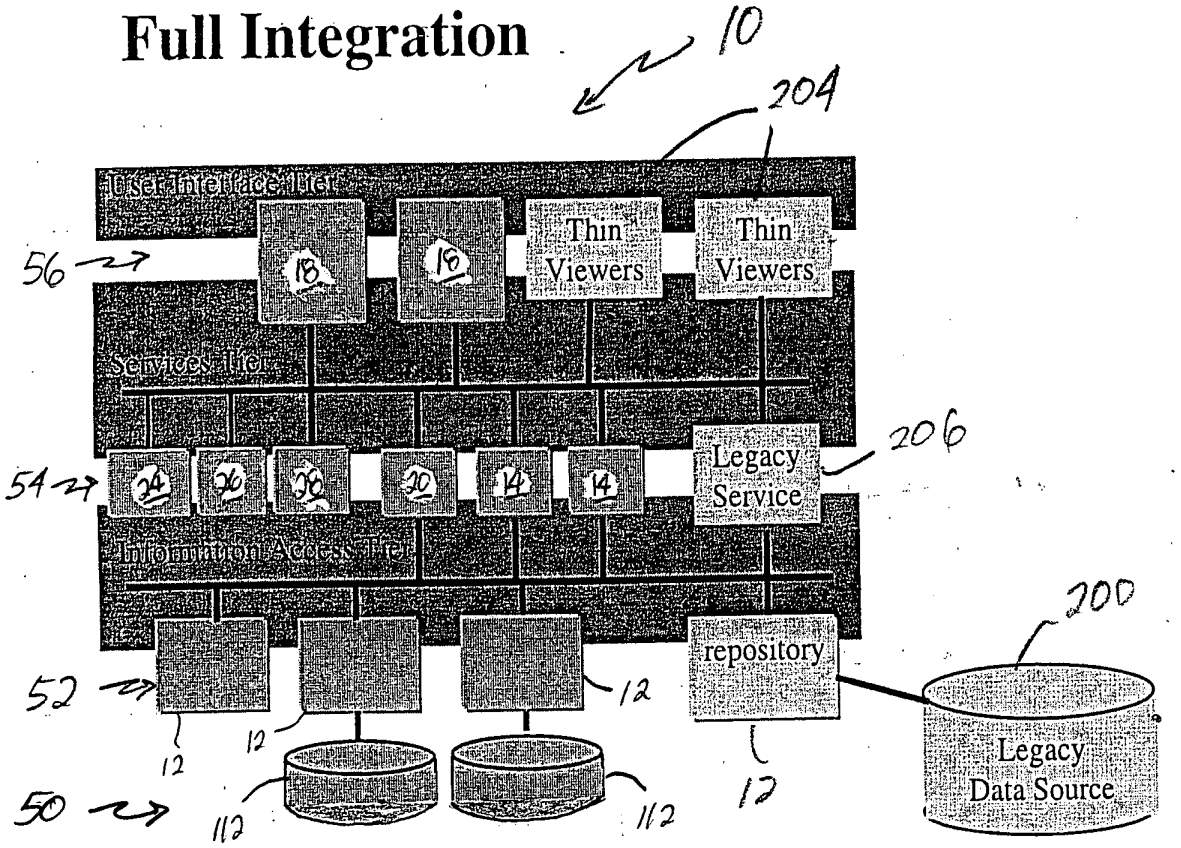
SYSTEM *Infrastructure* *Legacy System*

- Access data through Tsunami infrastructure
- Legacy viewers are now interact collaboratively
- Still maintain the option to interact directly with the data source
- Provides additional options for extending system capabilities

Fig. 12

Integration with legacy systems

Full Integration



SYSTEM *Infrastructure* *Legacy System*

- Rewrite viewers in Java making them web-enabled and machine independent
- Legacy processing becomes a system component available for enterprise usage
- Lowers maintenance cost
- Duplicate functionality removed across the enterprise
- Each enhancement is available to the entire enterprise

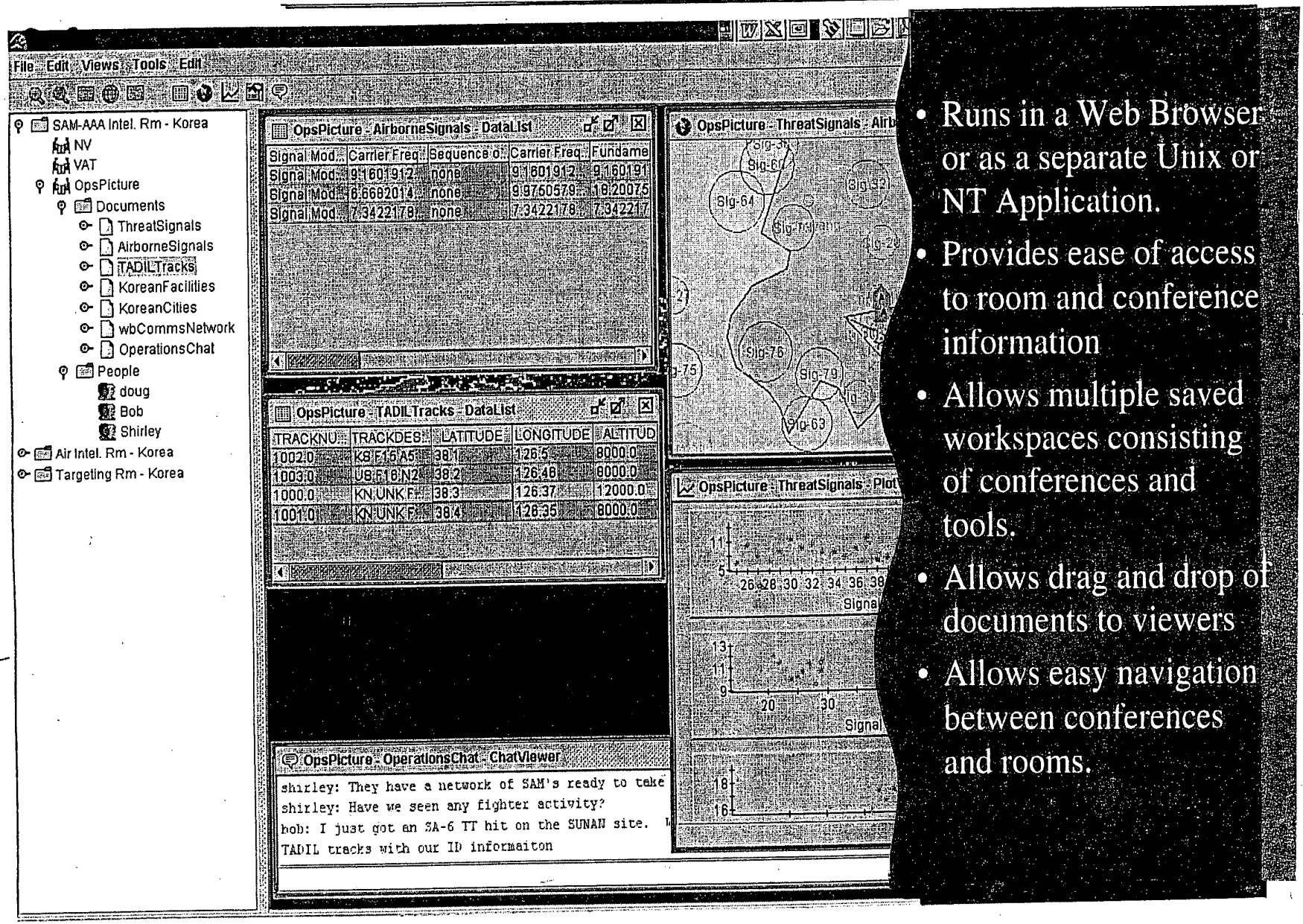
Fig. 13

Importance of Data-Centric Collaboration Framework

- Framework is applicable to most domains
- Small tools extend overall capability
  - Build domain or analyst specific tools--not systems
  - Adding single collaborative capabilities results in exponential growth of overall system capability
- Collaboration integral to framework
  - Instead of pasting images onto a whiteboard, collaborate on the tool itself using whiteboarding layer
  - No special logic needed in tools to support collaboration
- Supports legacy applications
  - Data is shared and not replicated, so changes to the data by legacy tools propagate to collaborative tools.

Fig. 14

Collaboration Application Management

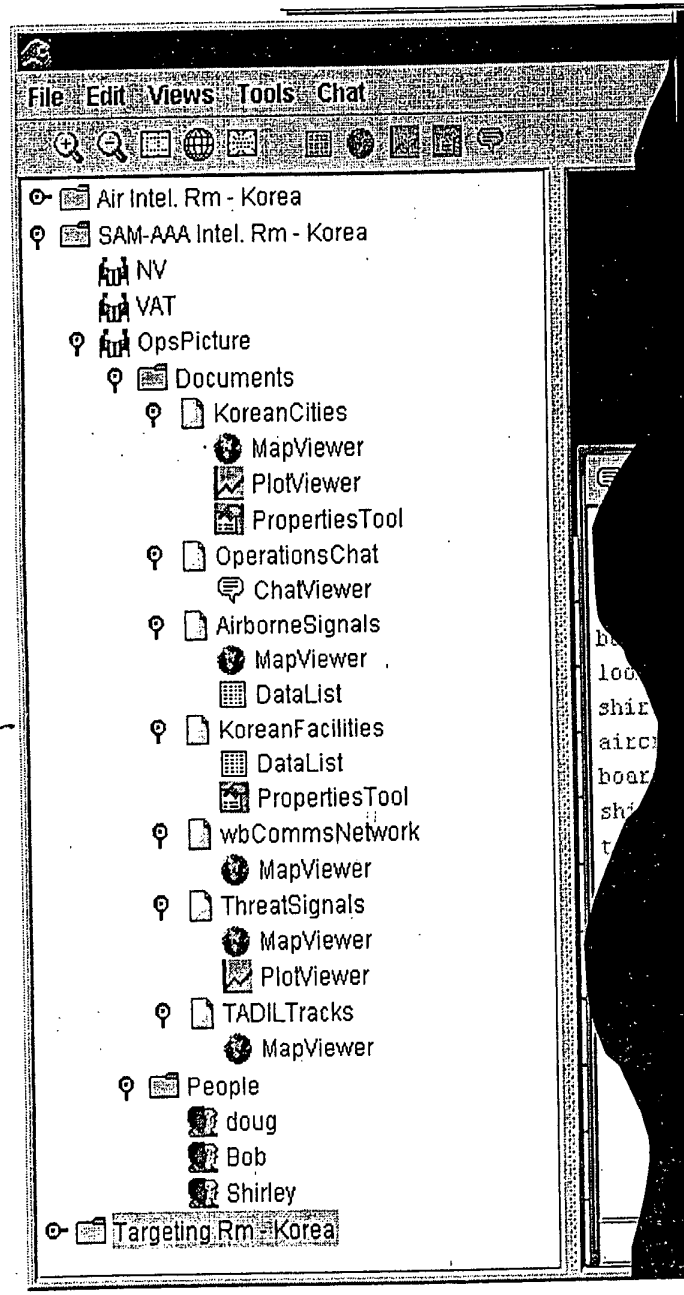


- Runs in a Web Browser or as a separate Unix or NT Application.
- Provides ease of access to room and conference information
- Allows multiple saved workspaces consisting of conferences and tools.
- Allows drag and drop of documents to viewers
- Allows easy navigation between conferences and rooms.

60

Fig. 15

Collaborative Application Management

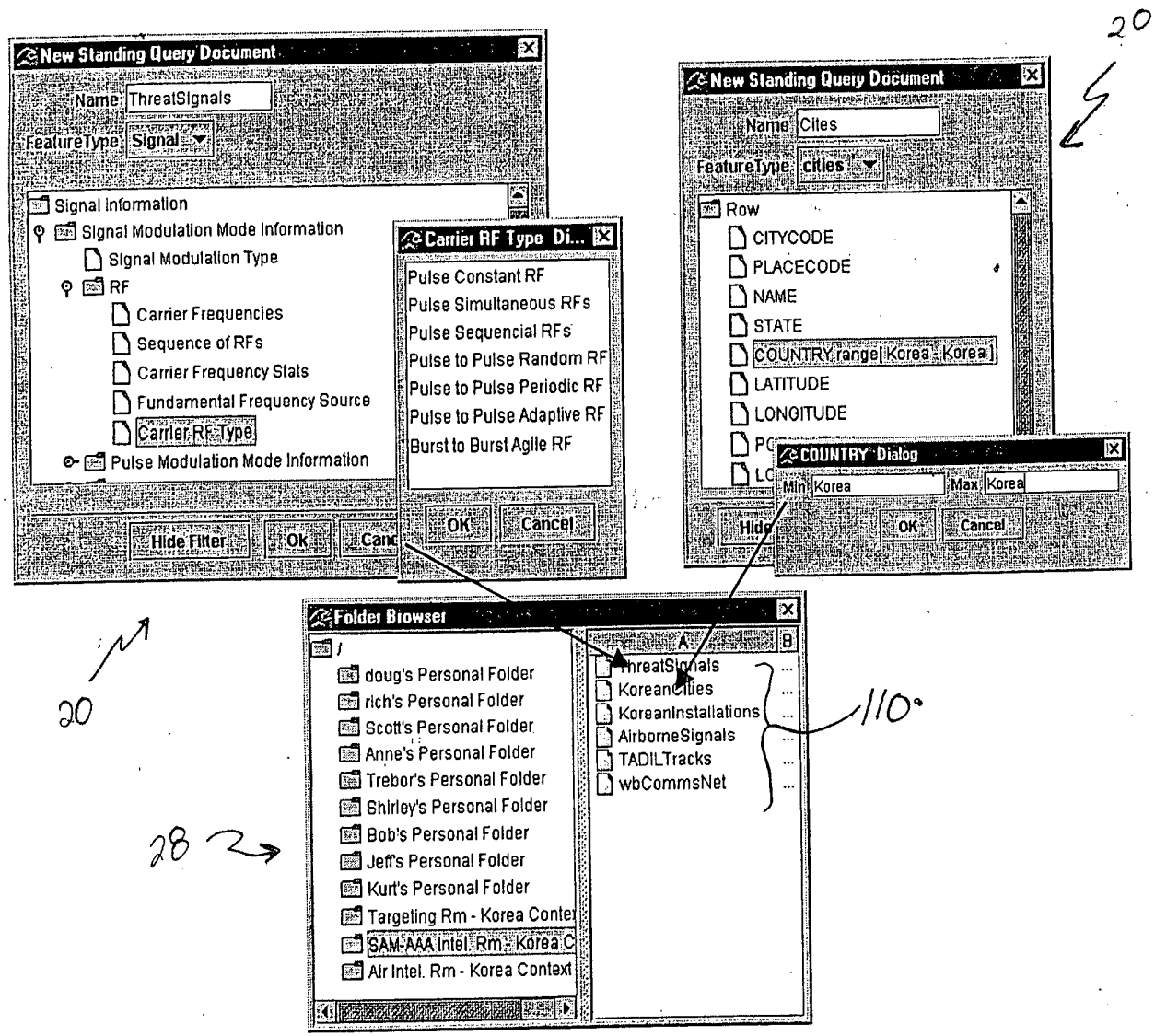


- Multiple Active Rooms/Places that contain Documents, People, Tools and Conferences
- Within a Conference a person or group of people analyze information and interact to solve problems
- People within a Conference are shown
- Documents being used within a Conference are shown
- Tools that are connected to documents are shown
- User can minimize/maximize/remove individual tools, tools associated with a conference or tools associated with a room
- Documents are dragged and dropped onto a conference from the document server window

60

Fig. 16

Dynamic Repository Query & Document Management

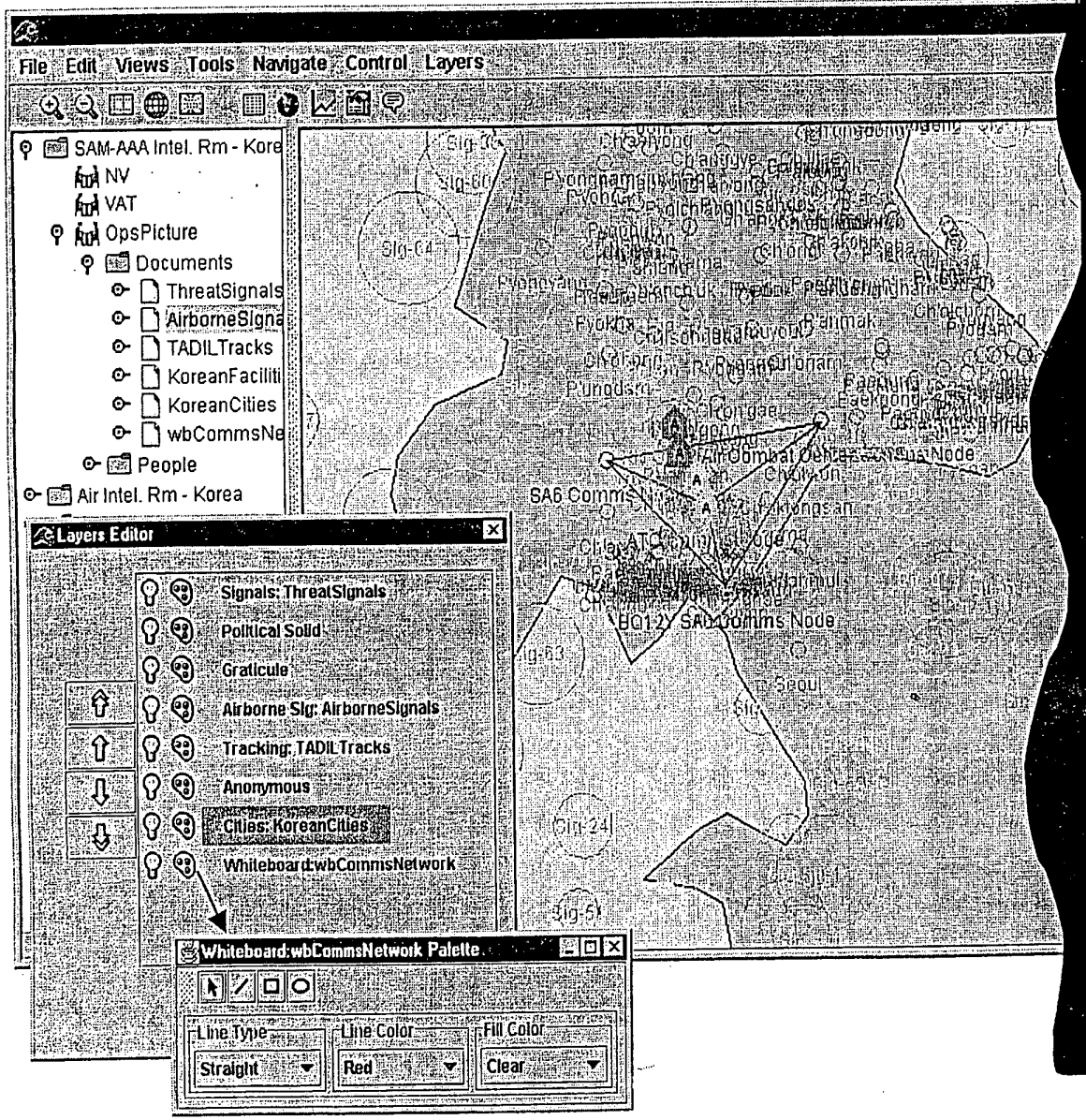


- Dynamically learns about repository
- Gets attribute meta-data from repository
- Creates agent representing standing query
- Results become a document which may be used for collaboration

Fig. 17

Map And White-Board Interaction

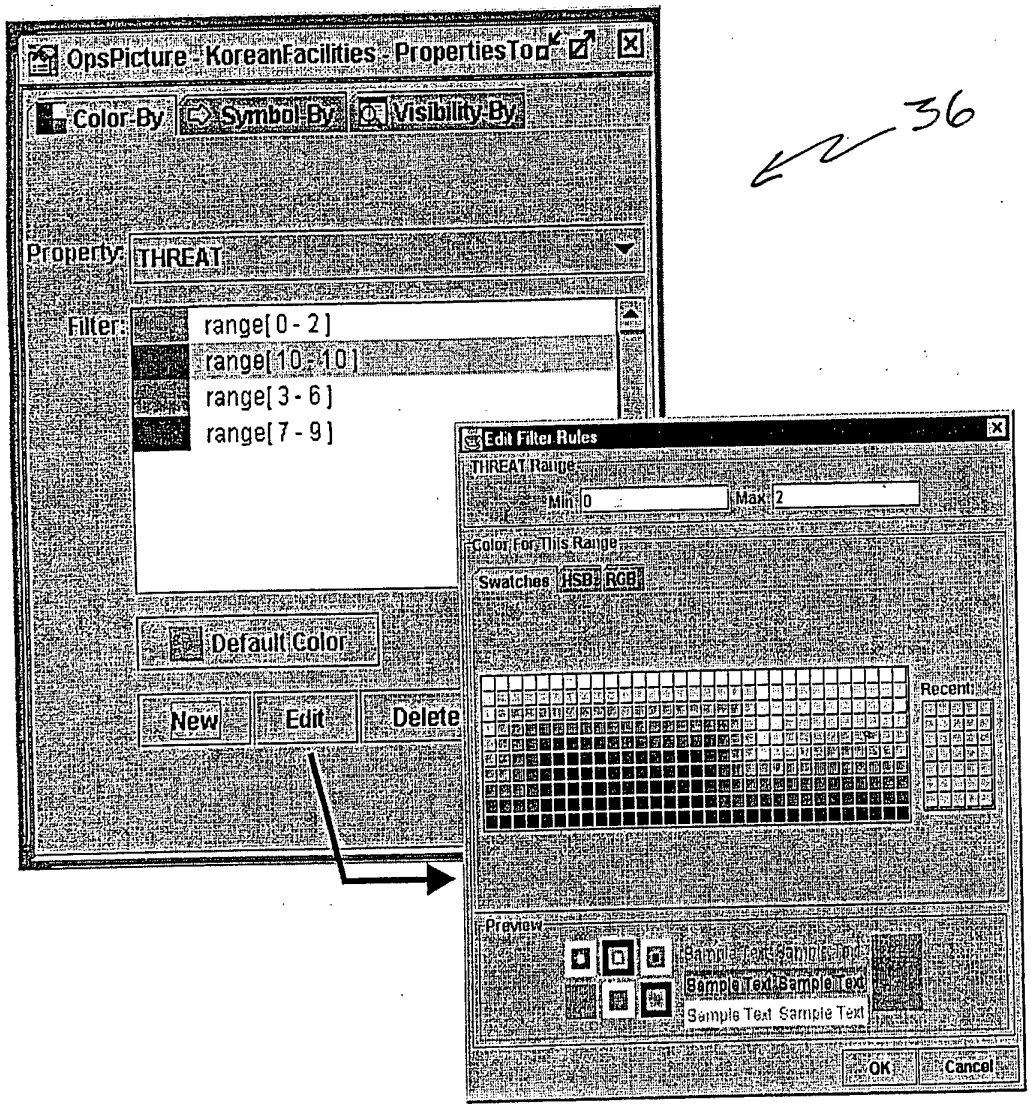
18



- The BBN Open Map Viewer was selected since it supported layering and a standards-based interface. No license fee is required. It is an Open Source component.
- A Mercator projection is shown with items colored via the data model
- A configurable pop-up menu can be seen
- Integration with commercial and legacy map products is based on OpenGIS standard APIs.

Fig. 18

Extended Properties Editor

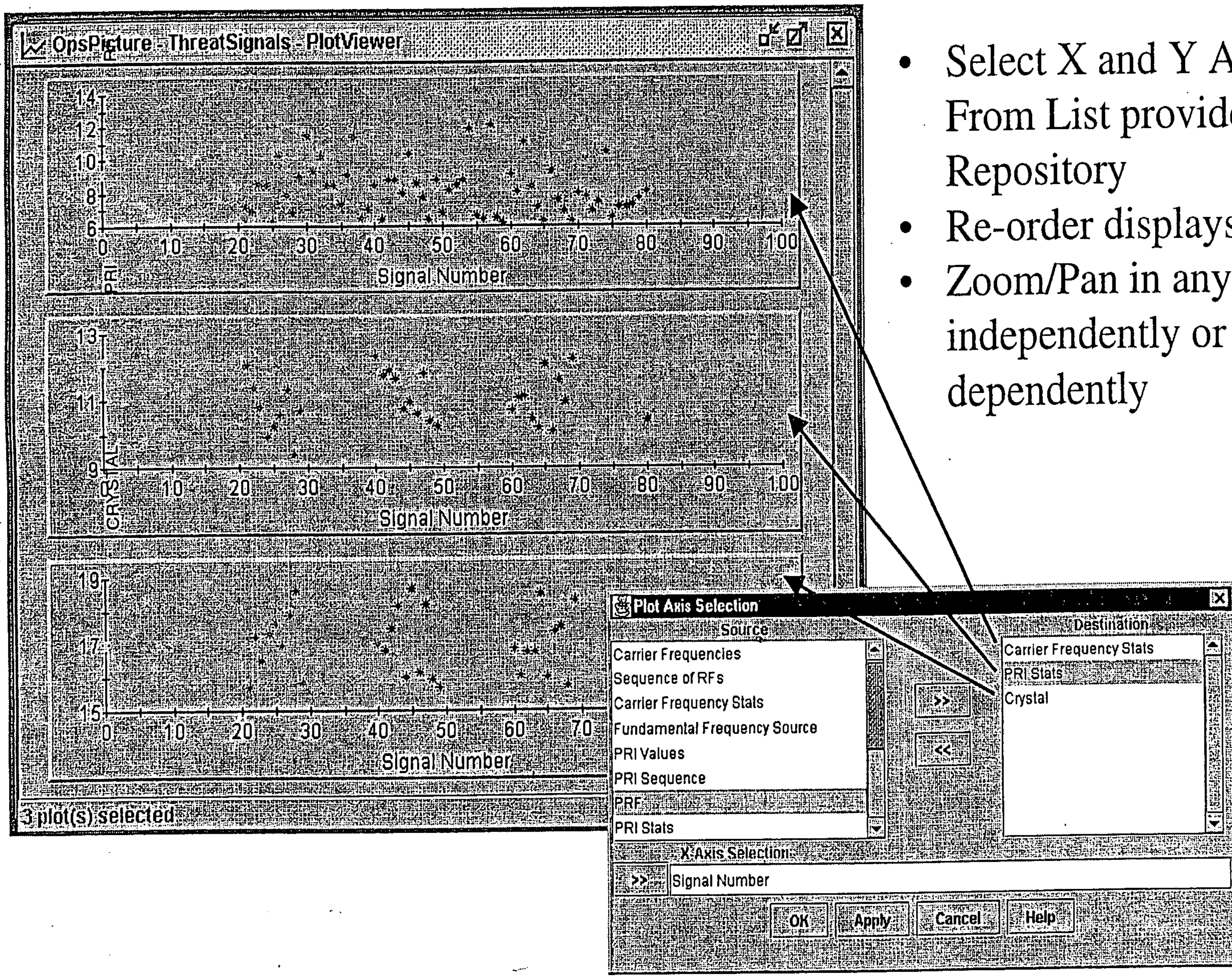


- Dynamically learns information schema from repository
- Attaches extended extended properties to data in the data channel
- Applied rules run as agents within the channel
- Extended Properties
  - Color
  - Highlight
  - Visibility
  - Label
  - Symbol
  - .....

Fig. 19

X-Y Plotter

18



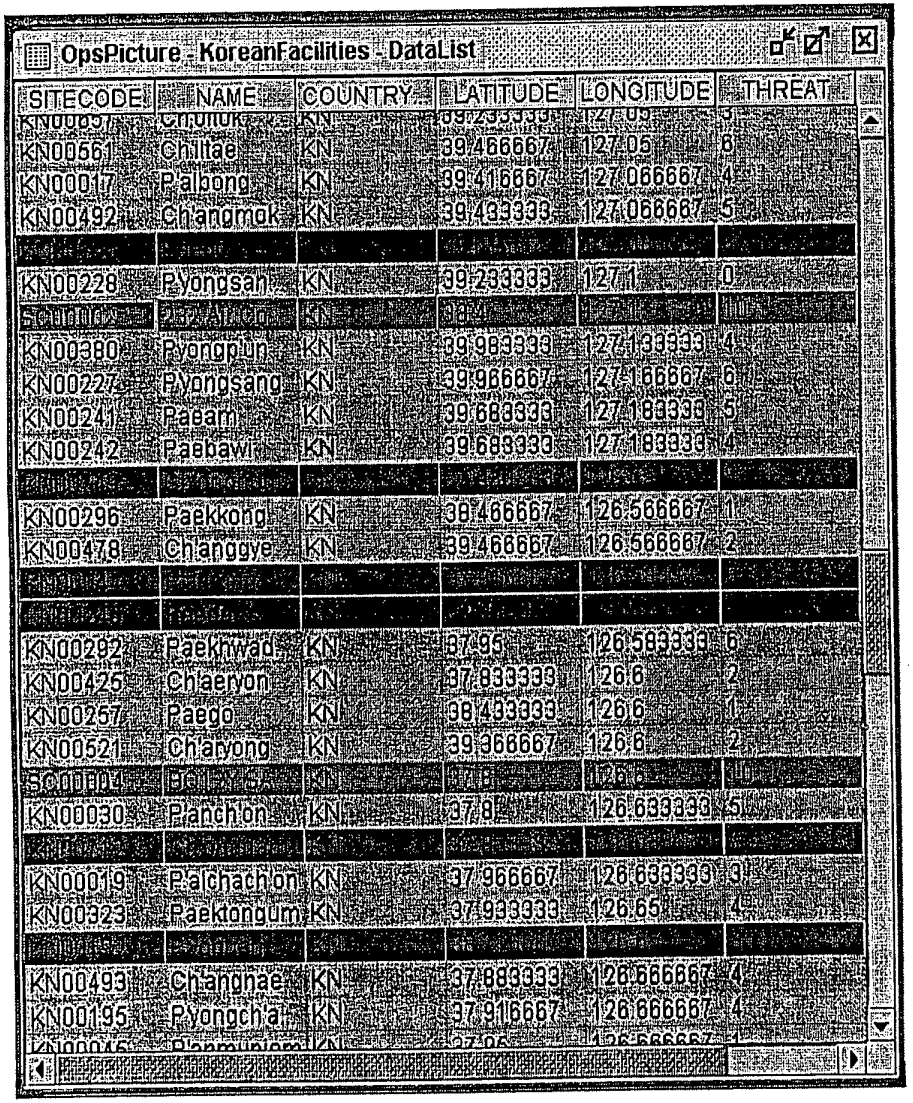
OpsPicture - ThreatSignals - PlotViewer
14
12
10
8
6
0 10 20 30 40 50 60 70 80 90 100
PRI
Signal Number
13
11
9
CRYSTAL
0 10 20 30 40 50 60 70 80 90 100
Signal Number
19
17
15
0 10 20 30 40 50 60 70
Signal Number
3 plot(s) selected
Plot Axis Selection
Source
Carrier Frequencies
Sequence of RFs
Carrier Frequency Stats
Fundamental Frequency Source
PRI Values
PRI Sequence
PRF
PRI Stats
Destination
Carrier Frequency Stats
PRI Stats
Crystal
>>
<<
X-Axis Selection
>>
Signal Number
OK
Apply
Cancel
Help

- Select X and Y Attributes From List provided by Repository
- Re-order displays
- Zoom/Pan in any display independently or dependently

Fig. 20

18

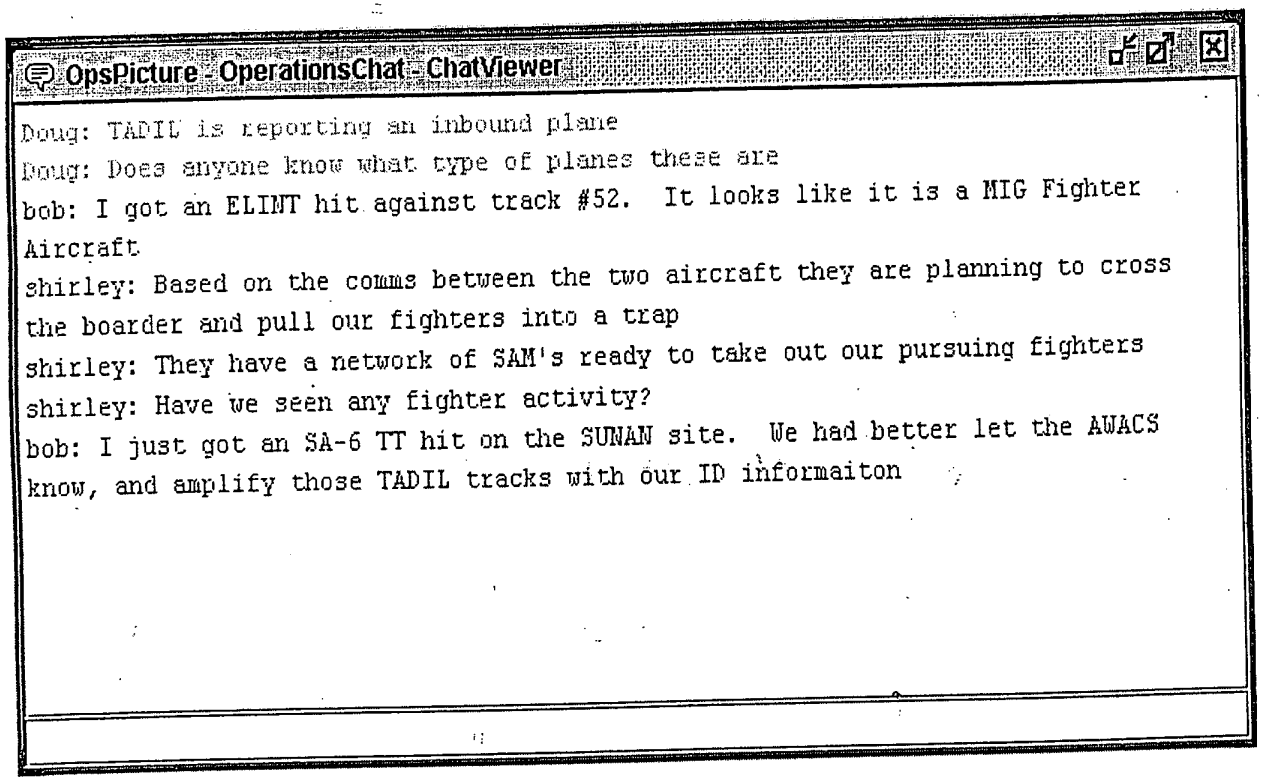
List Viewer



- Sorting
- Row Selection
- Row Coloring
- Row Hiding
- Choose Attributes to View

Fig. 21

Chat Tool



- Chat supports multi-user conversations from multiple conferences in multiple contexts
- People connect to a document and communicate
- People in the same conference see the same visualization properties like color and visibility of participants inputs
- Conversations are persistent over time

Fig. 22

Performance Metrics

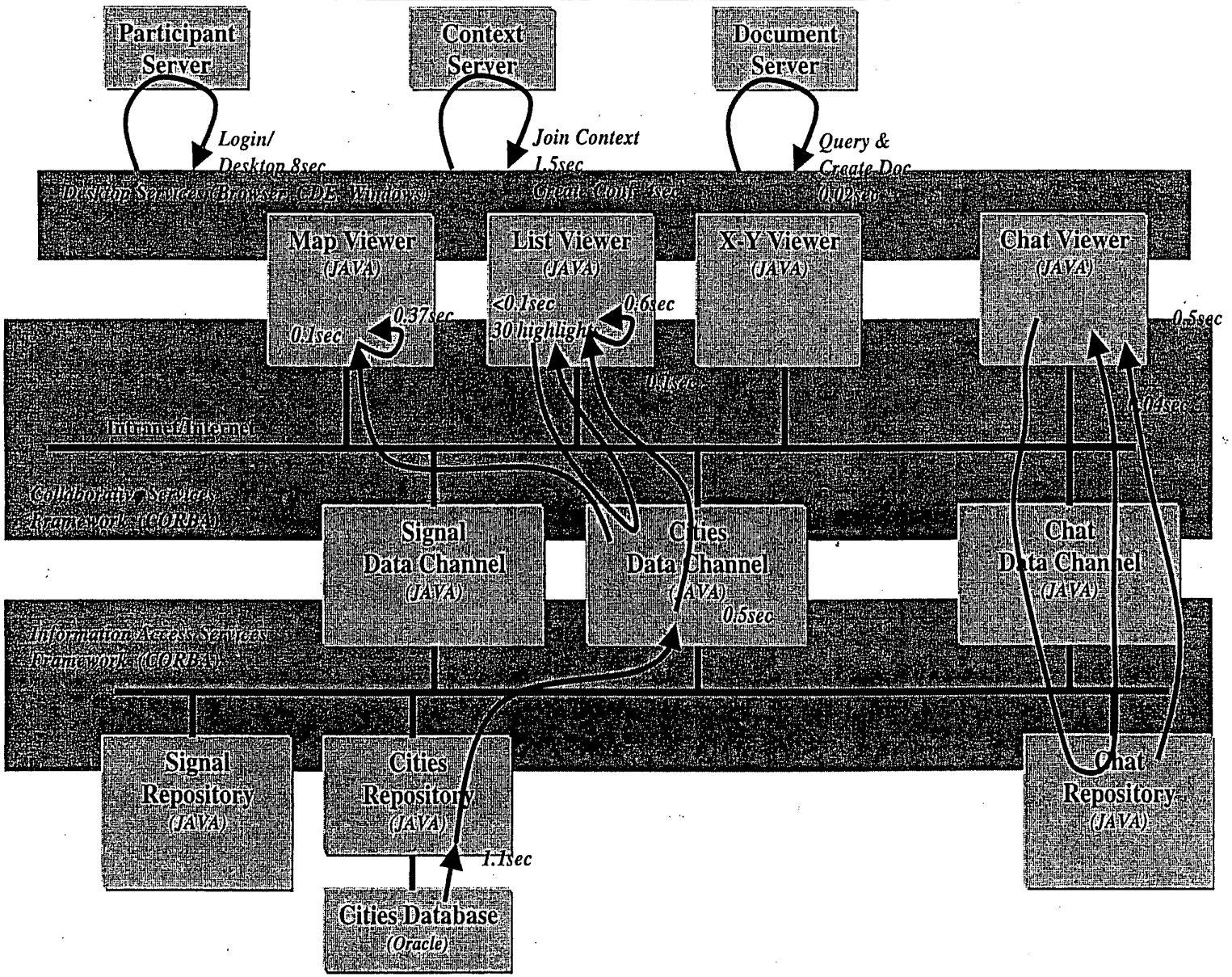
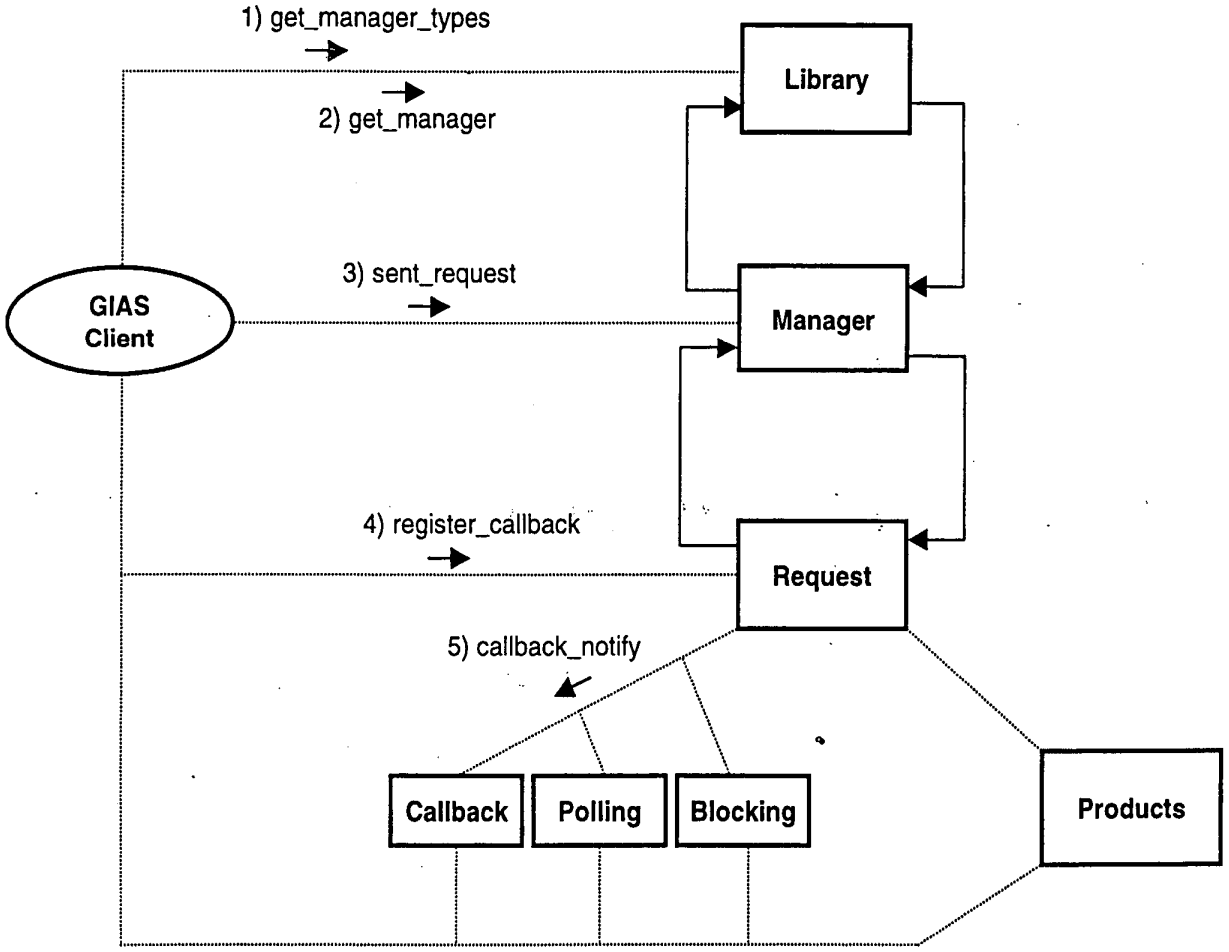


Fig. 23

USIGS - Geospatial and Imagery Access Services Specification



USIGS
Common Object Specification
(UCOS)

- Create Request
- Submit Query Request
- Submit Standing Query Request

- Dynamic discovery of information sources
- Dynamic discovery of access techniques
- Synchronous, Asynchronous, Polling Access mechanisms
- Clients autonomous request executing within the data environment
- All Interfaces and Structures represented within IDL (UCOS - DAG)

Fig. 24

Information Access Services

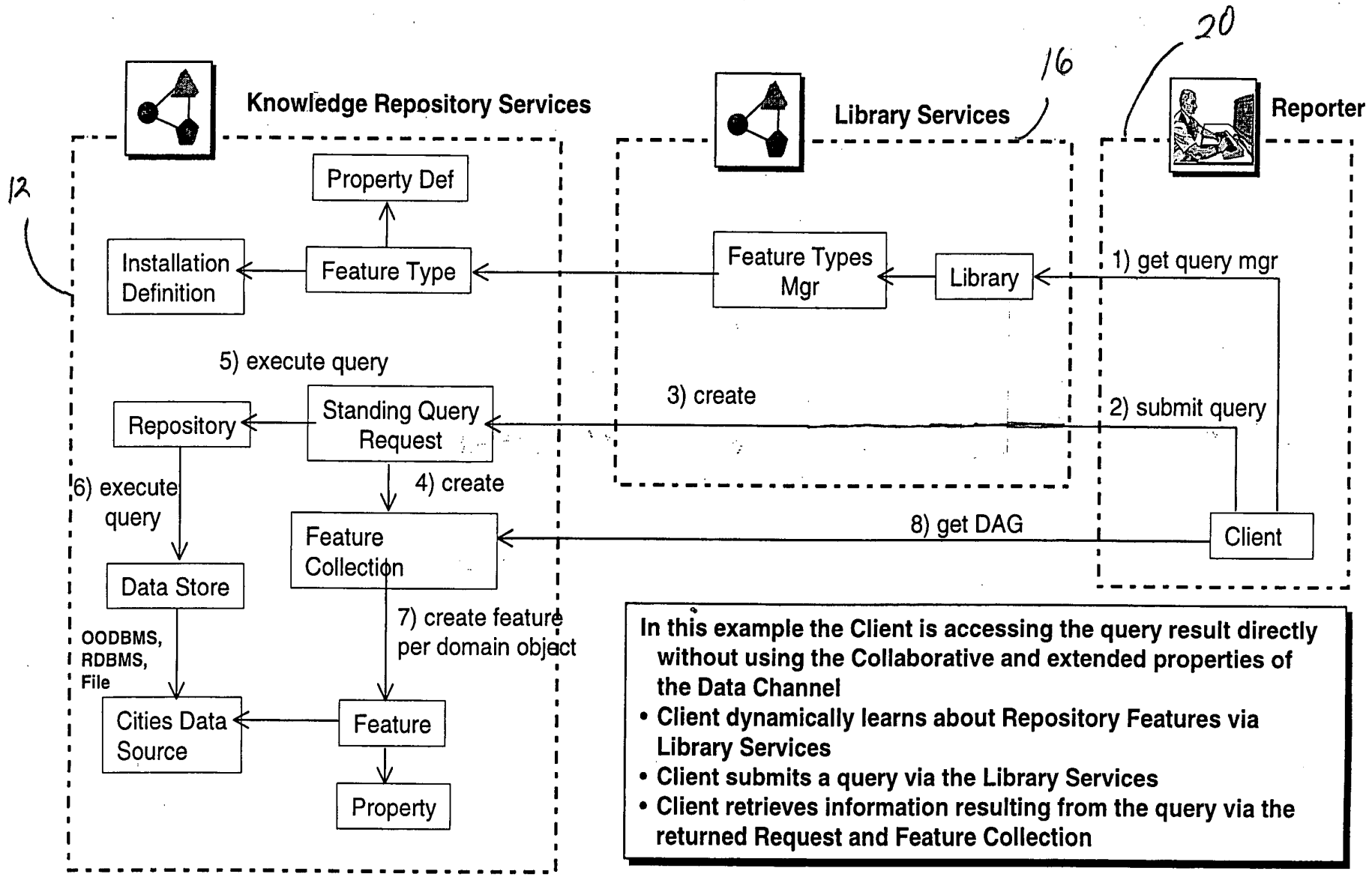


Fig. 25

Library Virtual Access

Requesting Information

- Client knows only about Library
- Client learns about Feature Types through Feature Type Mgr
- Client submits query through the Standing Query Mgr
- Repository and Feature Collection adapt to Database Particulars

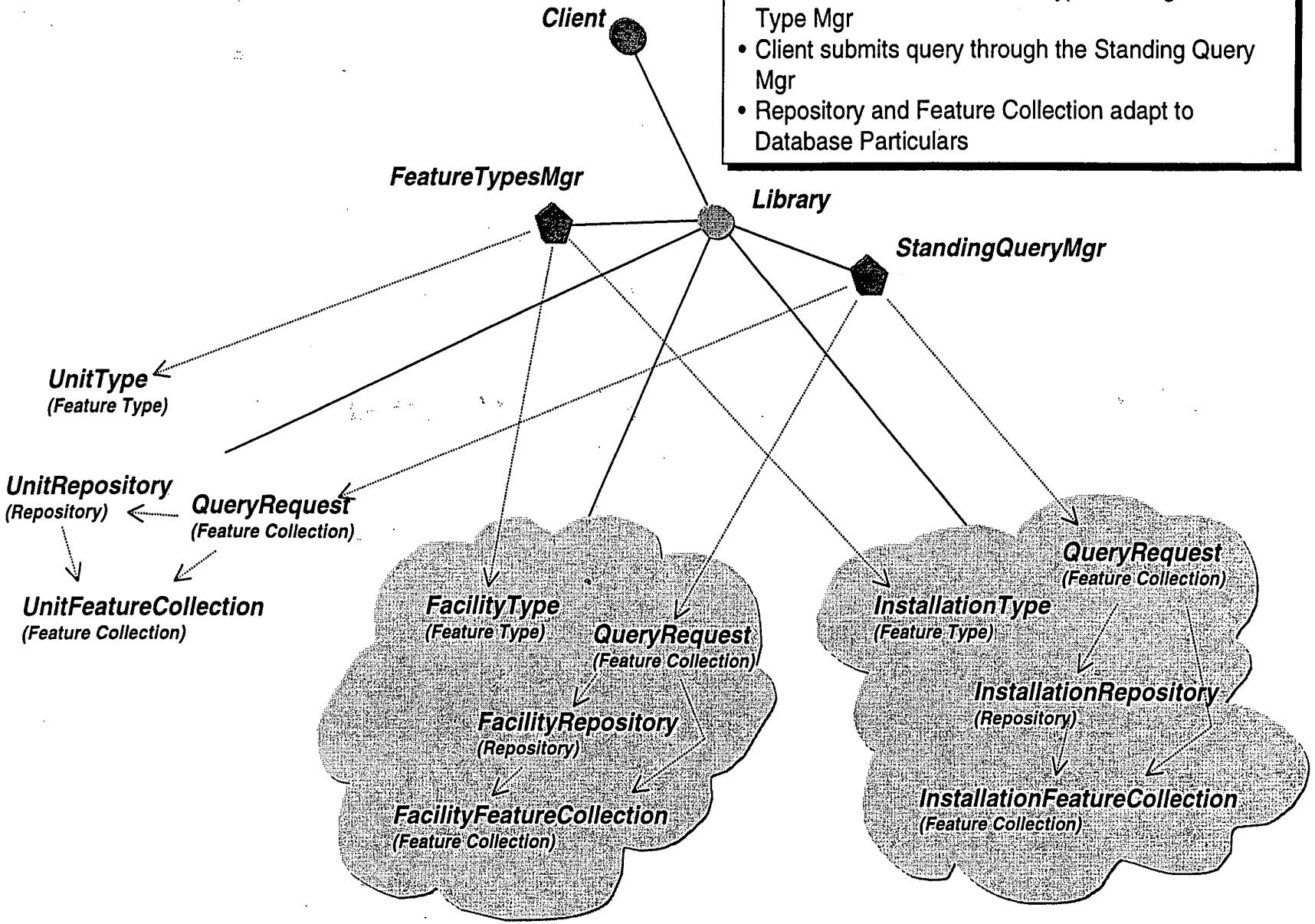
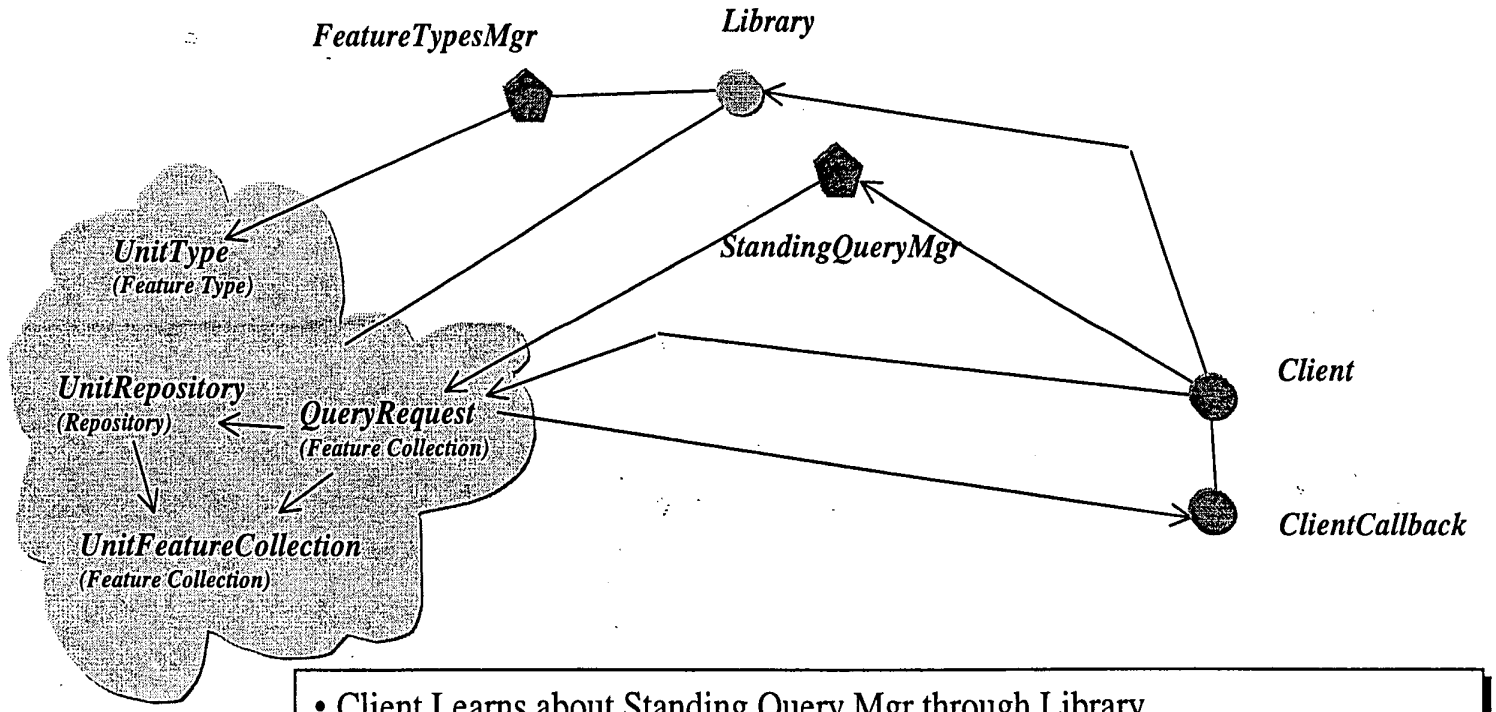


Fig. 26

Library Virtual Access

Accessing Current Information



- Client Learns about Standing Query Mgr through Library
- Query Manager returns a reference to a Request Object for each client query method invocation
- Client interacts with Request for Query Control and Status
- Request supports Synchronous, Polling, and A-Synchronous Client interfaces. ClientCallback is used for A-Synchronous feedback on query state

Fig. 27

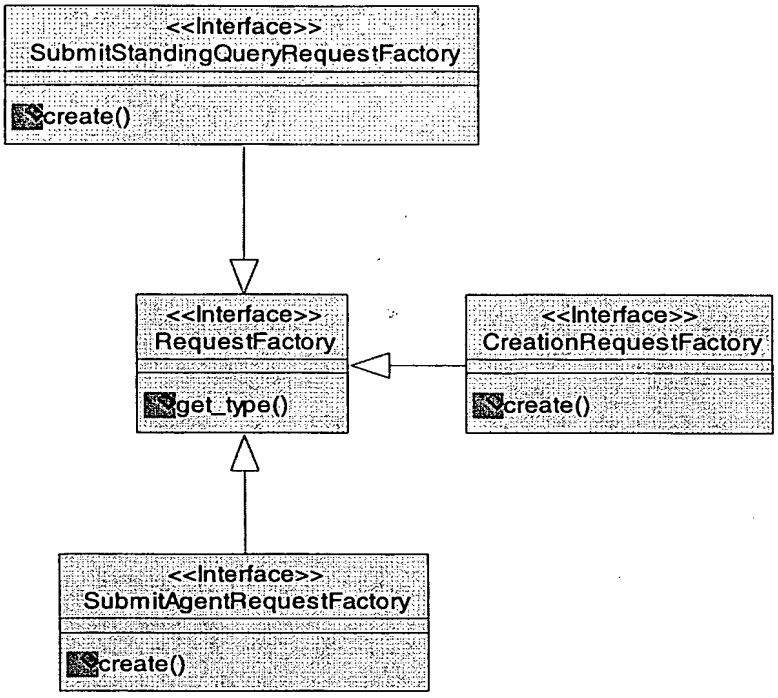

<<Interface>>
SubmitStandingQueryRequestFactory
create()
<<Interface>>
RequestFactory
get_type()
<<Interface>>
CreationRequestFactory
create()
<<Interface>>
SubmitAgentRequestFactory
create()

Fig. 28

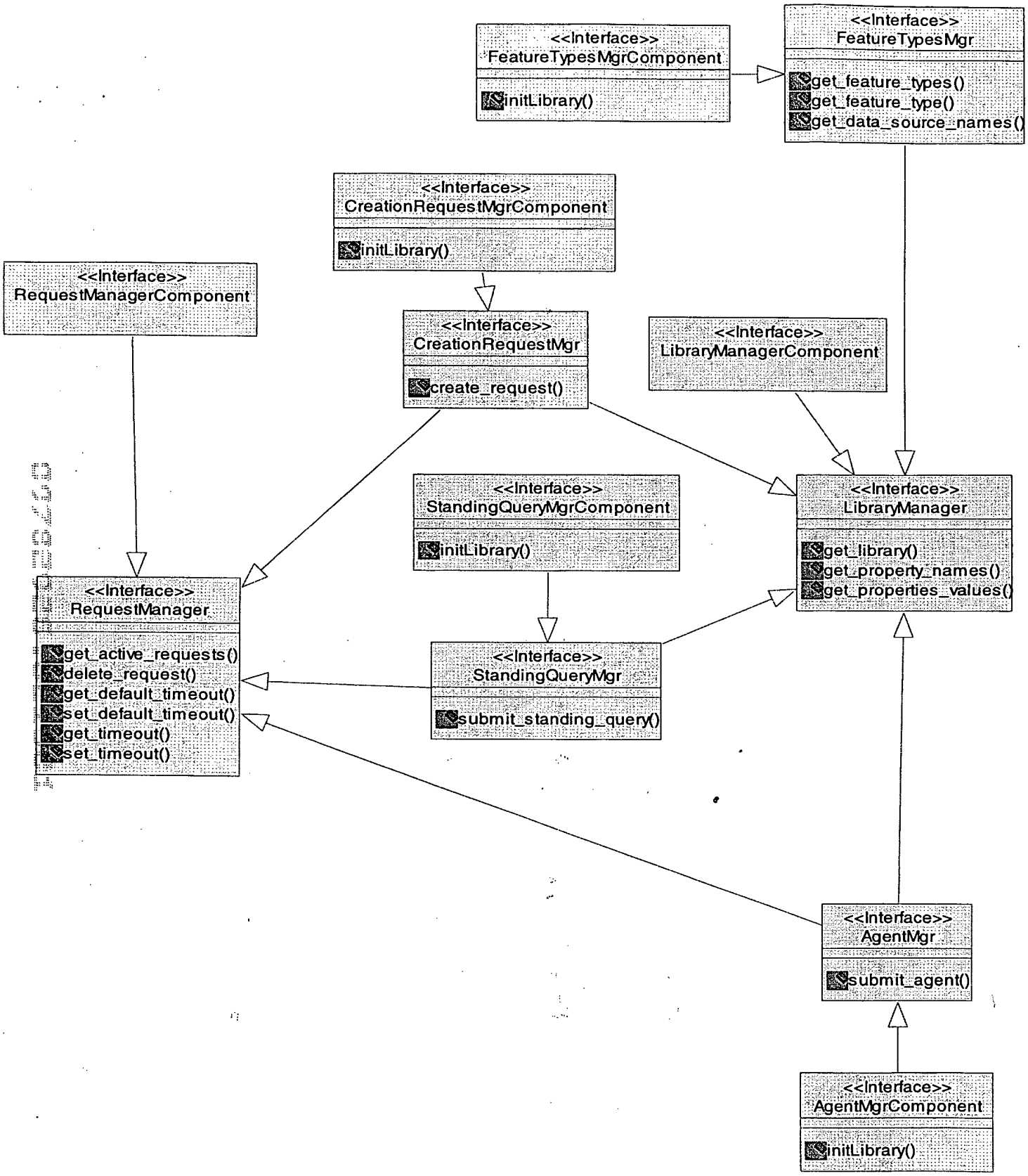

<<Interface>>
FeatureTypesMgrComponent
initLibrary()
<<Interface>>
FeatureTypesMgr
get_feature_types()
get_feature_type()
get_data_source_names()
<<Interface>>
CreationRequestMgrComponent
initLibrary()
<<Interface>>
RequestManagerComponent
<<Interface>>
CreationRequestMgr
create_request()
<<Interface>>
LibraryManagerComponent
<<Interface>>
StandingQueryMgrComponent
initLibrary()
<<Interface>>
LibraryManager
get_library()
get_property_names()
get_properties_values()
<<Interface>>
RequestManager
get_active_requests()
delete_request()
get_default_timeout()
set_default_timeout()
get_timeout()
set_timeout()
<<Interface>>
StandingQueryMgr
submit_standing_query()
<<Interface>>
AgentMgr
submit_agent()
<<Interface>>
AgentMgrComponent
initLibrary()

Fig. 29

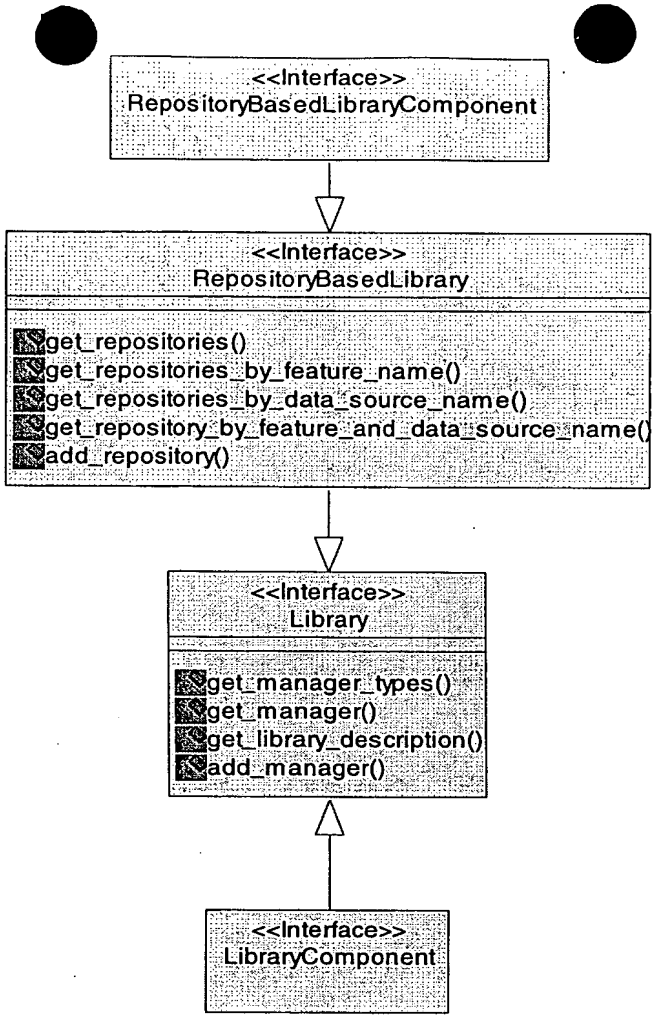

<<Interface>>
RepositoryBasedLibraryComponent
<<Interface>>
RepositoryBasedLibrary
get_repositories()
get_repositories_by_feature_name()
get_repositories_by_data_source_name()
get_repository_by_feature_and_data_source_name()
add_repository()
<<Interface>>
Library
get_manager_types()
get_manager()
get_library_description()
add_manager()
<<Interface>>
LibraryComponent

Fig. 30

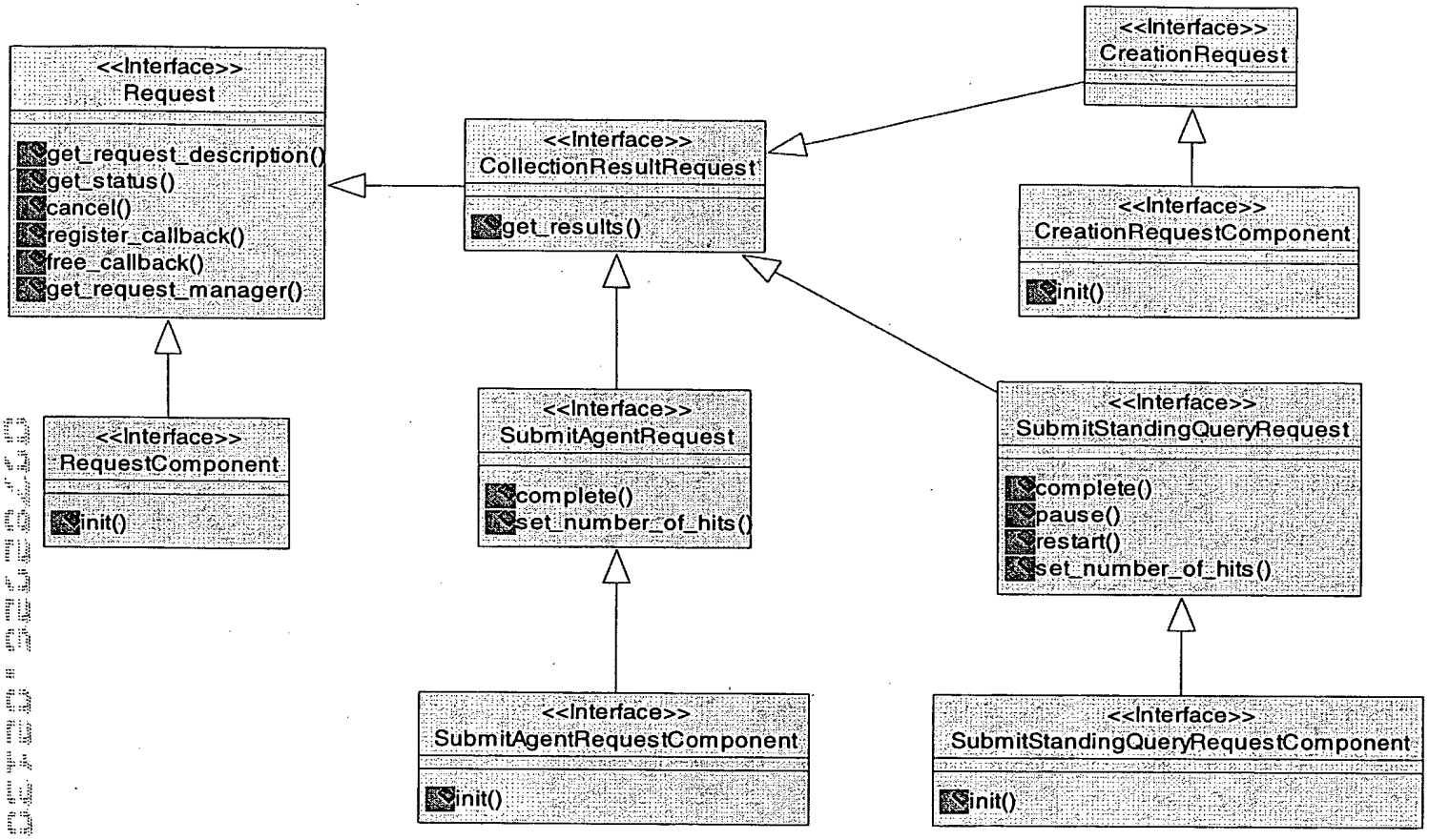

<<Interface>>
Request
get_request_description()
get_status()
cancel()
register_callback()
free_callback()
get_request_manager()
<<Interface>>
CreationRequest
<<Interface>>
CollectionResultRequest
get_results()
<<Interface>>
CreationRequestComponent
init()
<<Interface>>
RequestComponent
init()
<<Interface>>
SubmitAgentRequest
complete()
set_number_of_hits()
<<Interface>>
SubmitStandingQueryRequest
complete()
pause()
restart()
set_number_of_hits()
<<Interface>>
SubmitAgentRequestComponent
init()
<<Interface>>
SubmitStandingQueryRequestComponent
init()

Fig. 31

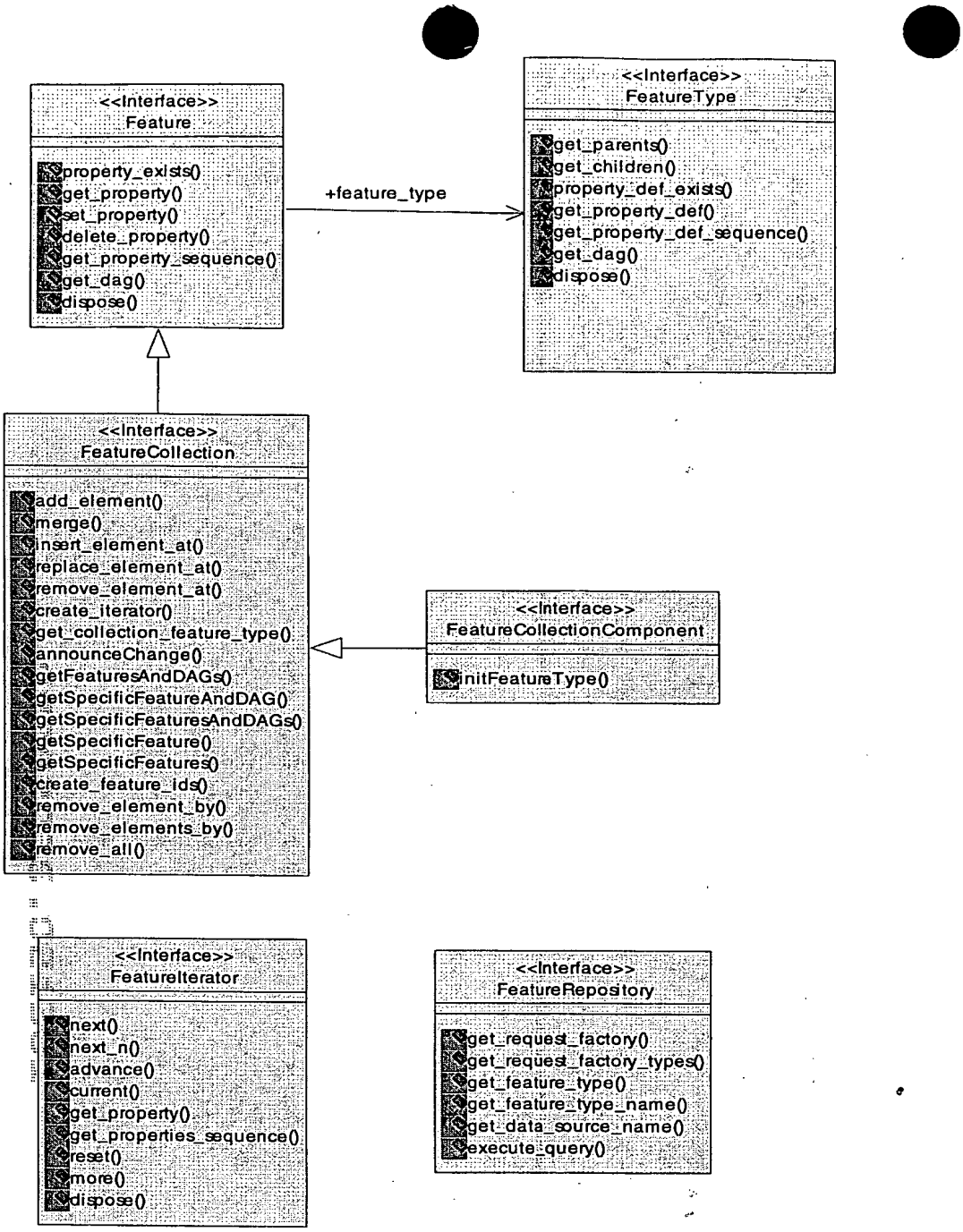

<<Interface>>
Feature
property_exists()
get_property()
set_property()
delete_property()
get_property_sequence()
get_dag()
dispose()
+feature_type
<<Interface>>
FeatureType
get_parents()
get_children()
property_def_exists()
get_property_def()
get_property_def_sequence()
get_dag()
dispose()
<<Interface>>
FeatureCollection
add_element()
merge()
insert_element_at()
replace_element_at()
remove_element_at()
create_iterator()
get_collection_feature_type()
announceChange()
getFeaturesAndDAGs()
getSpecificFeatureAndDAG()
getSpecificFeaturesAndDAGs()
getSpecificFeature()
getSpecificFeatures()
create_feature_ids()
remove_element_by()
remove_elements_by()
remove_all()
<<Interface>>
FeatureCollectionComponent
initFeatureType()
<<Interface>>
FeatureIterator
next()
next_n()
advance()
current()
get_property()
get_properties_sequence()
reset()
more()
dispose()
<<Interface>>
FeatureRepository
get_request_factory()
get_request_factory_types()
get_feature_type()
get_feature_type_name()
get_data_source_name()
execute_query()

Fig. 32

Data Channel Services Framework

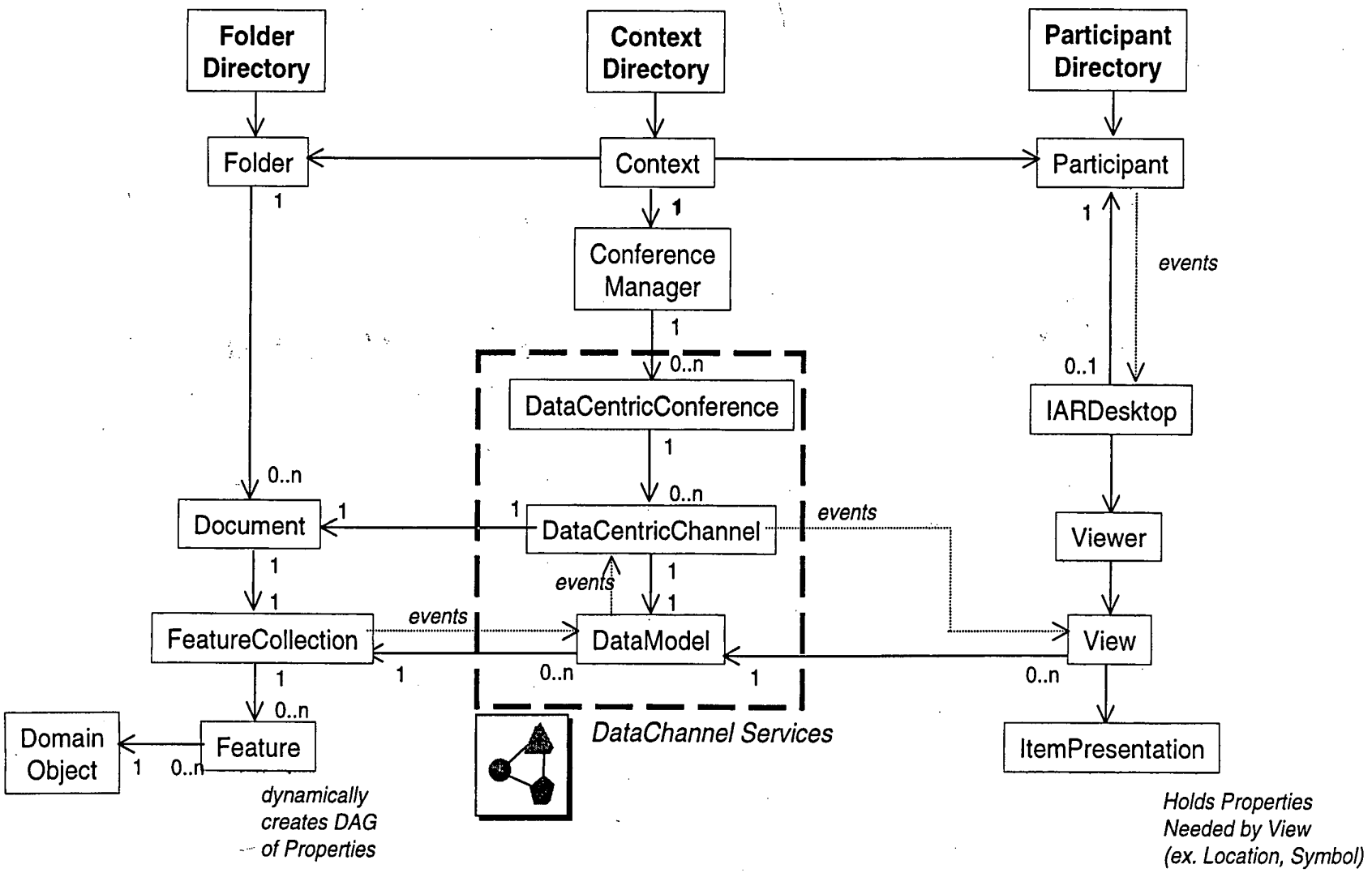


Fig. 33

Versioning Data Changes in the Data Channel

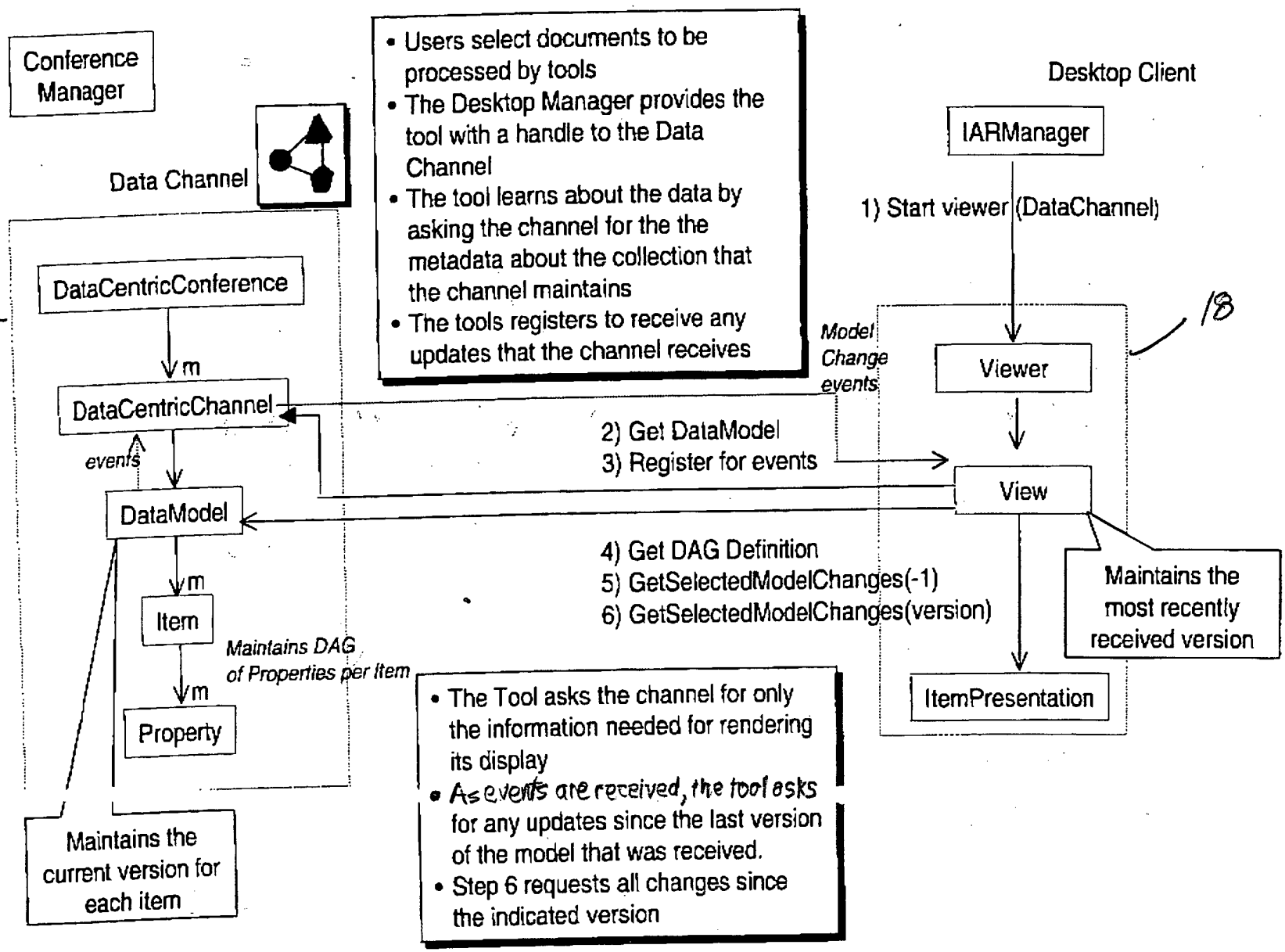


Fig. 34

OpenGIS Simple Features Specification
Understanding a Feature Collection

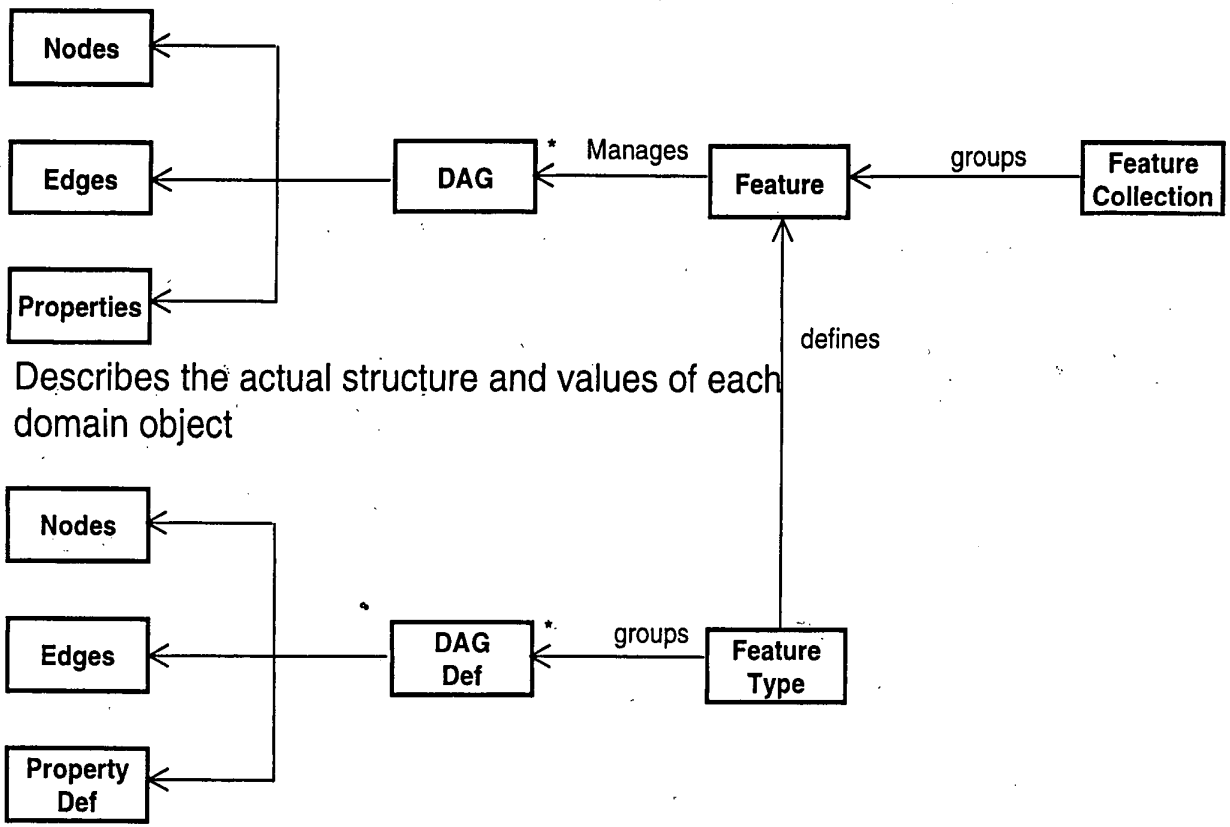
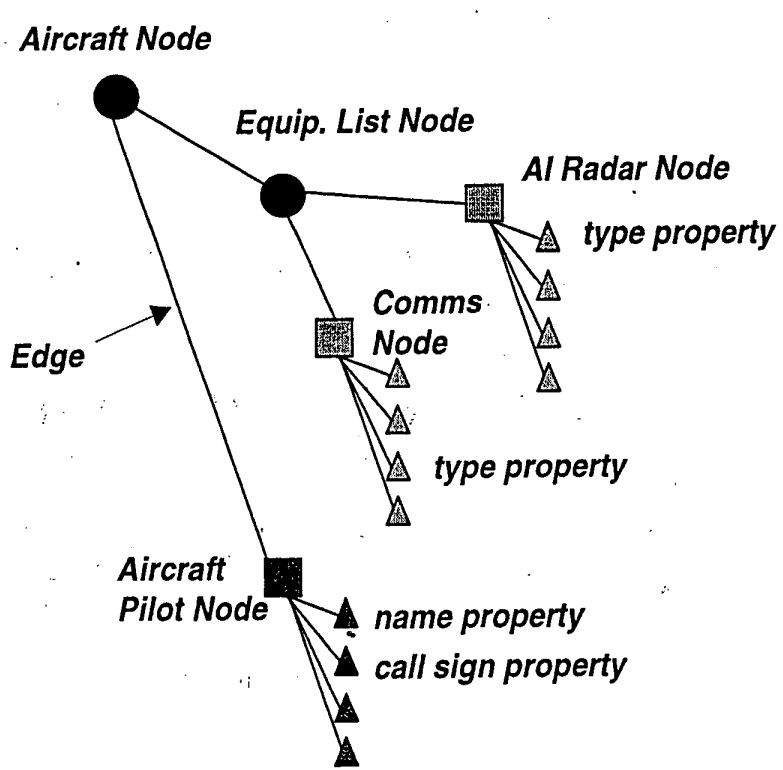


Fig. 35

Directed A-Cyclic Graph (DAG)



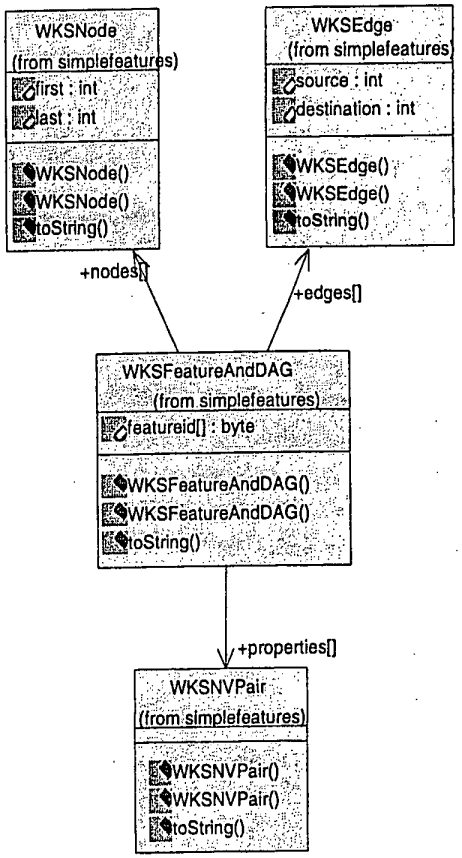
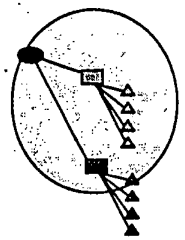


Fig. 36

Directed A-Cyclic Graph (DAG)



node #1 = name:"Airplane"; type:"element"
property #1 = "name": (SimpleFeatures.WKSString)"Airframe 321"
property #2 = "type": (SimpleFeatures.WKSString)"Mig29"

edge #2
source node = #1
destination node = #3

edge #1
source node = #1
destination node = #2

node #3 = name:"EquipmentList"; type:"nodelist"

node #2 = name: "Pilot"; type:"element"
property #4 = "start time": (SimpleFeatures.WKSAbsTime) 12/1/99 12:32:12.1222
property #5 = "stop time": (SimpleFeatures.WKSAbsTime) 12/1/99 12:32:23.233
property #6 = "name": (SimpleFeatures.WKSString) "Joe Smith"
property #7 = "all sign": (SimpleFeatures.WKSString) "idaho"

edge #3
source node = #3
destination node = #4

edge #4
source node = #3
destination node = #5

node #4 = name: "AI Radar"; type:"element"
property #11 = "rf range": (SimpleFeatures.WKSDoubleRange) 10.0 - 14.0
property #12 = " bandwidth":(SimpleFeatures.WKSDouble) 1.0
property #13 = "pw range":(SimpleFeatures.WKSDoubleRange) 0.1-100.0
property #14 = "polarization type":(SimpleFeatures.WKSCategory) "all"
property #15 = "sensitivity":(SimpleFeatures:WKSCategory) "sidelobe"
property #16 = "modulation on pulse":(SimpleFeatures:WKSCategory) "IMOP", "UMOP"
property #17 = "dwell duration": (SimpleFeatures.WKSTime) 00:00:00.1000
property #18 = "data type": (SimpleFeatures.WKSCategory) "pdw"
property #19 = "aircraft side": (SimpleFeatures.WKSCategory) "left"
property #20 = "location":(SimpleFeatures.WKSBoolean) yes

IDL DAG Structure Passed

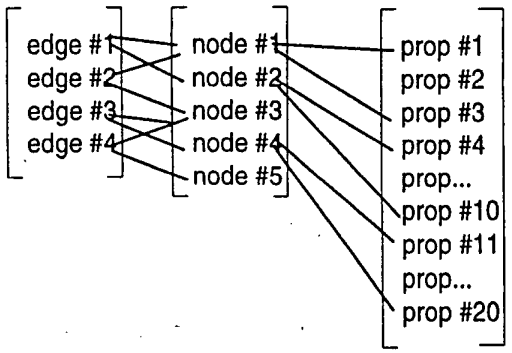


Fig. 37